МИНОБРНАУКИ РОССИИ

ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ «ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ЭКОНОМИКИ И СЕРВИСА» В Г. АРТЕМЕ

УТВЕРЖДАЮ
Заместитель директора
филиала
_____ О.И. ИВАНЮГА

РАБОЧАЯ УЧЕБНАЯ ПРОГРАММА ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

ПМ.01 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ

ПРОГРАММЫ ПОДГОТОВКИ СПЕЦИАЛИСТОВ СРЕДНЕГО ЗВЕНА
по специальности
Программирование в компьютерных системах

Уровень подготовки: базовый

Год набора на ООП
2020

Артем 2020

Рабочая программа профессионального модуля ПМ.01 Разработка программных модулей программного обеспечения для компьютерных систем разработана в соответствии с Разъяснениями по формированию примерных программ начального профессионального и среднего профессионального образования на основе Федеральных государственных образовательных стандартов НПО и СПО, утвержденными Департаментом государственной политики и нормативно - правового регулирования в сфере образования Минобрнауки РФ от 27 августа 2009 года, с учетом требований Федерального государственного образовательного стандарта среднего профессионального образования (далее – СПО), утвержденного приказом Минобрнауки РФ от 28 июля 2014 г. № 804 для освоения программы подготовки специалистов среднего звена по специальности **09.02.03 Программирование в компьютерных системах,** реализуемой колледжем Филиала федерального государственного бюджетного образовательного учреждения высшего образования «Владивостокский государственный университет экономики и сервиса» в г. Артеме (далее Филиал ФГБОУ ВО «ВГУЭС» в г. Артеме).

Разработчик:

Место работы	Занимаемая должность, ученая степень и ученое (почетное) звание, квалификационная категория	Инициалы, фамилия	Подпись
Филиал ФГБОУ ВО «ВГУЭС» в г.Артёме	Преподаватель кафедры экономики, управления и информационных технологий	Е.В.Волошин	

Эксперты:

Место работы	Занимаемая должность, ученая степень и ученое (почетное) звание, квалификационная категория	Инициалы, фамилия	Подпись
Филиал ФГБОУ ВО «ВГУЭС» в г. Артеме	Руководитель информационно-технического центра	В.В. Неслюзов	
ООО «СКС – Сервис», г. Артем	Директор	О.В. Бажин	

Рабочая программа профессионального учебного модуля ПМ.01 Разработка программных модулей программного обеспечения для компьютерных систем

ОДОБРЕНА:
на заседании кафедры экономики, управления и информационных технологий
филиала ФГБОУ ВО «ВГУЭС» в г. Артёме
Протокол № 14 от 06 мая 2020 года.

И.о.зав. кафедрой ЭУИТ _____ А.А.Власенко

СОГЛАСОВАНА:
Заведующий отделением _____ М.С.Словикова
Методист
учебно-методической части _____ Т.И.Теплякова

2

СОДЕРЖАНИЕ

1. ПАСПОРТ РАБОЧЕЙ ПРОГРАММЫ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

1.1. Область применения рабочей учебной программы

Рабочая программа профессионального модуля **ПМ.01. Разработка программных модулей программного обеспечения компьютерных систем** является частью программы подготовки специалистов среднего звена (ППССЗ) по специальности 09.02.03 Программирование в компьютерных системах, разработанной в соответствии с ФГОС СПО по специальности в части освоения основного вида профессиональной деятельности (ВПД) **Разработка программных модулей программного обеспечения компьютерных систем** и соответствующих профессиональных компетенций (ПК):

ПК 1.1 Выполнять разработку спецификаций отдельных компонент.
ПК 1.2 Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.
ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств.
ПК 1.4 Выполнять тестирование программных модулей.
ПК 1.5 Осуществлять оптимизацию программного кода модуля.
ПК 1.6 Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.

Данная программа профессионального модуля может быть использована в дополнительном профессиональном образовании (в программах повышения квалификации и переподготовки) и профессиональной подготовки.

1.2. Цели и задачи профессионального модуля – требования к результатам освоения профессионального модуля:

В результате освоения программы обучающийся должен:

иметь практический опыт:
- разработки алгоритма поставленной задачи и реализации его средствами автоматизированного проектирования;
- разработки кода программного продукта на основе готовой спецификации на уровне модуля;
- использования инструментальных средств на этапе отладки программного продукта;
- проведения тестирования программного модуля по определенному сценарию.

уметь:
- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля;
- оформлять документацию на программные средства;
- использовать инструментальные средства для автоматизации оформления документации.

знать:
- основные этапы разработки программного обеспечения;
- основные принципы технологии структурного и объектно-ориентированного программирования;
- основные принципы отладки и тестирования программных продуктов;
- методы и средства разработки технической документации.

При изучении дисциплины внимание студента будет обращено на её прикладной характер, на то, где и когда изучаемые теоретические положения, и практические навыки могут быть использованы в будущей практической деятельности.

1.3. Рекомендуемое количество часов на освоение учебной программы профессионального модуля:

всего – **456** часов, в том числе:

- максимальной учебной нагрузки обучающегося – **240** часов, включая:
- максимальной учебной нагрузки обучающегося из вариативной части -125 часов;
- обязательной аудиторной учебной нагрузки обучающегося – **160** часов,
- из них из вариативной части – 84часа;
- самостоятельной работы обучающегося – **80** часов, из них из вариативной части - 41 час;
- учебной практики – **108** часов;
- производственная практика по профилю специальности – **108** часов.

2. РЕЗУЛЬТАТЫ ОСВОЕНИЯ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

Результатом освоения учебной программы профессионального модуля является овладение обучающимися видом профессиональной деятельности **Разработка программных модулей программного обеспечения компьютерных систем**, в том числе профессиональными (ПК) и общими (ОК) компетенциями:

Код	Наименование результата обучения
ПК 1.1	Выполнять разработку спецификаций отдельных компонент.
ПК 1.2	Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.
ПК 1.3	Выполнять отладку программных модулей с использованием специализированных программных средств.
ПК 1.4	Выполнять тестирование программных модулей.
ПК 1.5	Осуществлять оптимизацию программного кода модуля.
ПК 1.6	Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.
ОК 1.	Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес
ОК 2.	Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.
ОК 3.	Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.
ОК 4.	Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.
ОК 5.	Использовать информационно-коммуникационные технологии в профессиональной депятельности.
ОК 6.	Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.
ОК 7.	Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.
ОК 8.	Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.
ОК 9.	Ориентироваться в условиях частой смены технологий в профессиональной деятельности.

3. СТРУКТУРА И СОДЕРЖАНИЕ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

3.1. Тематический план профессионального модуля

Коды профессиональных компетенций	Наименования разделов профессионального модуля	Всего часов	Объем времени, отведенный на освоение междисциплинарного курса					Практика	
			Обязательная аудиторная учебная нагрузка обучающегося			Самостоятельная работа обучающегося		Учебная, часов	Производственная (по профилю специальности), часов
			Всего, часов	в т.ч. лабораторные работы и практические занятия, часов	в т.ч. курсовая работа, часов	Всего, часов	в т.ч. курсовая работа, часов		
1	2	3	4	5	6	7	8	9	10
ПК 1.2-1.6	МДК 01.01. Системное программирование	135	90	46	-	45	-	108	-
ПК 1.1-1.5	МДК 01.02. Прикладное программирование	105	70	38	-	35	-	-	-
ПК 1.1–1.6	УП .01.01. Учебная практика	108							
ПК 1.1–1.6	ПП.01.01 Производственная практика (по профилю специальности)	108	-	-	-	-	-	-	108
	Всего:	456	160	84	-	80	-	108	108

Для характеристики уровня освоения учебного материала используются следующие обозначения:
1 – ознакомительный (узнавание ранее изученных объектов, свойств);
2 – репродуктивный (выполнение деятельности по образцу, инструкции или под руководством);
3 – продуктивный (планирование и самостоятельное выполнение деятельности, решение проблемных задач).

3.2. Содержание обучения по профессиональному модулю (ПМ.01)

Наименование разделов профессионального модуля (ПМ), междисциплинарного курса (МДК) и тем	Содержание учебного материала, лабораторные работы и практические занятия, самостоятельная работа обучающихся, курсовая работа	Объем часов	Уровень освоения
1	**2**	**3**	**4**
ПМ.01 Разработка программных модулей программного обеспечения компьютерных систем			
Раздел 1. МДК 01.01. Системное программирование (5 семестр)		**135**	
Тема 1.1 **Интерфейс Windows приложений.** **Осуществление разработки кода программного модуля на языке С++**			
	Содержание учебного материала: Введение. Понятие о программном модуле. Использование технологии объектно-ориентированного программирования для разработки интерфейсов пользователя.	**2**	**1,2**
	Внеаудиторная самостоятельная работа № 1 Подготовка реферата по одной из тем: 1.Новые концепции программирования. 2.Объектно-ориентированное программирование. 3.Каркас Windows приложений. 4. Методика создания программ.	**2**	**3**
	Содержание учебного материала: Взаимосвязь программирования с другими областями знаний. Взаимодействие с пользователем.	**2**	**1,2**
	Внеаудиторная самостоятельная работа № 2 Формирование программы, строящей прямоугольник в центре окна. Формирование программы движения шарика в окне.	**2**	**3**
	Содержание учебного материала: Стандартная заготовка Windows приложений.	**2**	**1,2**

	Внеаудиторная самостоятельная работа № 3 Формирование программы, которая строит окно как шахматную доску. Формирование программы движения шарика в окне.	**2**	**3**
	Содержание учебного материала: Разработка кода программного модуля на языке С++. Технология программирования.	**2**	**1,2**
	Внеаудиторная самостоятельная работа № 4 Создание текстовой программы вывода строки текста.	**2**	**3**
	Лабораторная работа № 1 по теме «Стандартная заготовка Windows приложений. Обработка сообщений. Создание программы по разработанному алгоритму»	**2**	**2,3**
	Лабораторная работа № 2 по теме «Разработка кода программного модуля с рисованием в окне геометрических фигур»	**2**	**2,3**
	Лабораторная работа № 3 по теме «Разработка кода программного модуля с выводом текста. Цвет, шрифт, фон»	**2**	**2,3**
	Лабораторная работа № 4 по теме «Разработка кода программного модуля с обработкой сообщений мыши»	**2**	**2,3**
	Лабораторная работа № 5 по теме «Разработка кода программного модуля с обработкой сообщений клавиатуры	**2**	**2,3**
	Лабораторная работа № 6 по теме «Разработка кода программного модуля, содержащего создание окон с полосами прокрутки»	**2**	**2,3**
Тема 1.2. **Технология работы с файлами при создании модуля**	**Содержание учебного материала:** Создание программы по разработанному алгоритму как отдельного программного модуля.	**2**	**1, 2**
	Содержание учебного материала: Разработка модуля, использующего организацию диалога выбора файлов.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 5. Создание программы, фильтрующей выбор имени файла.	**2**	**3**

	Содержание учебного материала: Панель инструментов. Чтение и запись файлов из библиотеки.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 6. Создание программы просмотра текстового файла с пунктом меню для чтения текстового файла в кодировке DOS и Windows.	**2**	**3**
	Содержание учебного материала: Основные принципы отладки и тестирования программного продукта.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 7 Создание программы, изменяющей активность пунктов меню.	**2**	**3**
	Лабораторная работа № 7 по теме «Выполнение отладки и тестирования программы на уровне модуля. Разработка кода программного модуля по организации диалога выбора файла»	**2**	**2, 3**
	Лабораторная работа № 8 по теме «Выполнение отладки и тестирования программы на уровне модуля. Чтение и запись файлов в библиотеке Win32 API».	**2**	**2, 3**
	Лабораторная работа № 9 по теме «Выполнение отладки и тестирования программы на уровне модуля. Разработка кода программного модуля с использованием меню»	**2**	**2, 3**
	Лабораторная работа № 10 по теме «Выполнение отладки и тестирования программы на уровне модуля. Разработка кода программного модуля с использованием панели инструментов»	**2**	**2, 3**
	Внеаудиторная самостоятельная работа № 8. Создание программы просмотра текстового файла с пунктом меню для записи текстового файла в кодировке DOS и Windows.	**2**	**3**
Тема 1.3 **Технология разработки модуля с использованием окон и элементов управления**	**Содержание учебного материала:** Основные этапы разработки программного обеспечения.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 9. Создание программы, изображающей шахматную доску, где каждая клетка будет дочерним окном.	**2**	**3**
	Содержание учебного материала:	**2**	**1, 2**

	Технология разработки модуля с использованием дочерних, всплывающих, диалоговых окон.		
	Внеаудиторная самостоятельная работа № 10. Создание программы построения графика, предусматривающей блокировку пункта меню до тех пор, пока не будет загружен файл с данными.	**2**	**3**
	Содержание учебного материала: Технология разработки модуля с использованием немодальных окон.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 11. Создание программы. В программе построения графика предусмотреть вывод координат во всплывающем окне.	**2**	**3**
	Содержание учебного материала: Технология разработки модуля с использованием стандартных диалоговых окон и элементов управления.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 12. Создание текстовой программы для работы со списком при помощи окна EditBox.	**2**	**3**
	Лабораторная работа № 11 по теме «Выполнение отладки и тестирования программы на уровне модуля. Основные этапы разработки программного обеспечения. Работы с окнами в приложении. Отладка модуля»	**2**	**2, 3**
	Лабораторная работа № 12 по теме «Тестирование элементов управления. Отладка модуля»	**2**	**2, 3**
	Лабораторная работа № 13 по теме **«**Разработка кода программного модуля с простым текстовым редактором на элементе управления EditBoxControl»	**2**	**2, 3**
Тема 1.4 Растровая графика. Разработка модуля с элементами графики	**Содержание учебного материала:** Растровая графика. Разработка модуля с элементами графики	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 13. Создание программы с функцией деления окна на четыре равные части и выводом в каждой части растровых изображений.	**2**	**3**

	Содержание учебного материала: Использование функций для создания программы с использованием графического образа.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 14. Создание программы - «просмотрщика» графических файлов.	**2**	**3**
	Содержание учебного материала: Разработка кода программного модуля с использованием метафайлов.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 15. Создание программы с использованием наложения изображений.	**2**	**3**
	Содержание учебного материала: Разработка кода программного модуля создания дискового файла.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 16. Создание программы с выбором шрифта в стандартном диалоговом окне и выводом текста по центру окна.	**2**	**3**
	Лабораторная работа №14 по теме «Разработка кода программного модуля с использованием функций создания графического образа. Отладка программы»	**2**	**2, 3**
	Лабораторная работа № 15 по теме «Разработка кода программного модуля с вращением графического образа. Отладка и тестирование модуля»	**2**	**2, 3**
	Лабораторная работа № 16 по теме «Разработка кода программного модуля с виртуальными окнами. Отладка и тестирование модуля»	**2**	**2, 3**
	Лабораторная работа № 17 по теме «Разработка кода программного модуля с растровым изображением в метафайле»	**2**	**2, 3**
Тема 1.5. Библиотека динамической компоновки DLL. Использование принципов структурного программирования	**Содержание учебного материала:** Библиотека динамической компоновки DLL.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 17. Создание DLL – библиотеки для работы с С-строкой, включив в неё аналоги стандартных функций.	**2**	**3**
	Содержание учебного материала: Использование принципов структурного программирования.	**2**	**1, 2**

	Внеаудиторная самостоятельная работа № 18. Построение демонстрационной задачи для использования библиотеки DLL при явном связывании.	**2**	**3**
	Содержание учебного материала: Основные принципы технологии структурного и объектно-ориентированного программирования. Создание и использование библиотеки DLL	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 19. Построение демонстрационной задачи для использования библиотеки DLL при неявном связывании.	**2**	**3**
	Содержание учебного материала: Библиотека общего использования. Явная и неявная загрузка ресурсов.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 20. Создание DLL-библиотеки с галереей рисунков одинакового размера.	**2**	**3**
	Лабораторная работа № 18 по теме «Основные принципы отладки и тестирования программных продуктов. Разработка кода программного модуля с использованием функций библиотеки DLL»	**2**	**2, 3**
	Лабораторная работа № 19 по теме «Разработка кода программного модуля с использованием неявного связывания»	**2**	**2, 3**
Тема 1.6 Методы и средства разработки технической документации программного продукта.	**Содержание учебного материала:** Методика и приёмы разработки технической документации программного продукта. Ориентация на пользователя.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 21. Подготовка реферата по теме Основные правила оформления программной документации.	**2**	**3**
	Содержание учебного материала: Подготовка документации на стадии разработки программного средства и на стадии формирования комплекта поставки. Стандартизация документации. Оформление документации в виде единого документа. Требования, предъявляемые к структуре документа.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 22. Подготовка реферата по теме	**2**	**3**

	Оформление текстового и графического материала.		
	Внеаудиторная самостоятельная работа № 23. Подготовка реферата по теме Виды программных документов.	**1**	**3**
	Лабораторная работа № 20 по теме «Проектирование структуры документа с технической документацией программного продукта. Этапы проектирования структуры документа»	**2**	**2, 3**
	Лабораторная работа № 21 по теме «Расположение материала. Типы информации и их компоновка. Разработка руководства системного программиста в текстовом редакторе»	**2**	**2, 3**
	Лабораторная работа № 22 по теме «Согласование терминологии предметной области. Согласование компьютерной терминологии. Разработка пояснительной записки в текстовом редакторе»	**2**	**2, 3**
	Лабораторная работа № 23 по теме «Создание руководства пользователя программного продукта с помощью текстового редактора и электронных таблиц» **Дифференцированный зачёт**	**2**	**2, 3**
Итого по МДК 01.01. Системное программирование в т. ч.: теоретическое обучение - лабораторные занятия - внеаудиторная самостоятельная работа		**135** **44** **46** **45**	
УП. 01.01. Учебная практика к МДК 01.01 Системное программирование (5 семестр)		**108**	
Тема 1. Вводное занятие	***Виды работ*** Инструктаж о прохождении практики. Знакомство с программой практики и порядок её проведения, изучение правил внутреннего распорядка, знакомство с графиком работы студентов, ведения дневника практики, составление отчета. Инструктаж по технике безопасности, пожаробезопасности, производственной санитарии под роспись в журнале. Правила безопасности при работе с компьютером.	**6**	**2, 3**

Тема 2. Разработка алгоритма поставленной задачи и реализация его средствами автоматизированного проектирования.	***Виды работ*** Анализ поставленной задачи. Выбор методов и разработка основных алгоритмов решения задачи. Разработка технического задания. Примерные темы заданий для выполнения практической работы: – Обработка сообщений – Рисование геометрических фигур в окне – Вывод текста – Диалог с пользователем – Чтение и запись файлов в библиотеке Win32 API – Диалоговые окна – Растровая графика – Анимация – Библиотеки динамической компоновки DLL	**6**	**2, 3**
	Виды работ Разработка структуры и конкретных компонент разрабатываемого программного обеспечения, в том числе схемы алгоритмов, их общее описание. Определение свойств входных и выходных данных поставленной задачи.	**6**	**2, 3**
	Виды работ Анализ процесса обработки информации и выбор структур данных для её хранения	**6**	**2, 3**
Тема 3.Разработка кода программного продукта на основе готовой спецификации на уровне модуля.	***Виды работ*** Выбор технологии и среды программирования. Проектирование интерфейса пользователя. Проектирование классов предметной области. Организация обработки сообщений.	**6**	**2, 3**
	Виды работ Использование директив препроцессора. Организация диалога с пользователем. Тестирование элементов управления.	**6**	**2, 3**
	Виды работ Работа с панелями инструментов**.** Разработка форм ввода-вывода информации. Чтение и запись файлов в библиотеке. Организация работы с файлами.	**6**	**2, 3**

	Виды работ Разработка кода программного продукта на языке С++ на уровне модуля.	**6**	**2, 3**
Тема 4. Использование инструментальных средств на этапе отладки программного продукта.	***Виды работ*** Выбор стратегии тестирования и разработка тестов. Отладка кода программного продукта, используя возможности отладчика.	**6**	**2, 3**
	Виды работ Использование средств отладки, предоставляемых интерфейсом пользователя. Определение мест программы, в которых необходимо установить точки останова.	**6**	**2, 3**
	Виды работ Использование команд меню Debug, Go для анализа значения переменных**.**	**6**	**2, 3**
Тема 5. Проведение тестирования программного модуля по определенному сценарию.	***Виды работ*** Ручной контроль программного модуля. Проверка структуры программного модуля.	**6**	**2, 3**
	Виды работ Структурное тестирование. Устранение утечки памяти. Исследование возможных причин утечки памяти.	**6**	**2, 3**
	Виды работ Функциональное тестирование. Особенности отладки приложений, использующих шаблоны функций и классов.	**6**	**2, 3**
	Виды работ Оценочное тестирование программного продукта. Анализ соответствия разработанного программного продукта постановке задачи.	**6**	**2, 3**
Тема 6. Оформление документации на программные средства.	***Виды работ*** Составление программной документации. Определение сведений, необходимых для сопровождения и эксплуатации программного продукта. Разработка пояснительной записки, содержащей информацию о структуре и конкретных компонентах программного обеспечения, в том числе схемы алгоритмов, их общее описание, обоснование принятых технических решений.	**6**	**2, 3**
	Виды работ Разработка руководства системного программиста, содержащего сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения.	**6**	**2, 3**

	Виды работ Разработка руководства по техническому обслуживанию, содержащего сведения для применения тестовых и диагностических программ при обслуживании технических средств.	**6**	**2, 3**
Раздел 2 МДК.01.02. Прикладное программирование (6 семестр)		**105**	
Тема 1. Основные возможности комплексной среды разработки приложений Qt			
Тема 1.1Технология программирования на Qt. Установка среды Qt на ПК пользователя.	**Содержание учебного материала:** **Введение.** Понятие о программном модуле. Разработка кода программного модуля. Компонент программного модуля. Технология программирования. Взаимосвязь программирования с другими областями знаний. Взаимодействие с пользователем. Компоновка виджетов. Использование справочной литературы.	**2**	**1,2**
	Внеаудиторная самостоятельная работа № 1 Подготовка реферата по одной из тем: 1.Новые концепции программирования. 2.Объектно-ориентированное программирование. 3.Однофайловые программы.	**2**	**3**
	Лабораторная работа № 1 по теме «Установка среды Qt на ПК. Разработка кода программного модуля»	**2**	**2,3**
Тема 1.2. Создание диалоговых окон	**Содержание учебного материала:** Создание программы по разработанному алгоритму как отдельного модуля. Подклассы QDialog.Технология сигналов и слотов.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 2 Подготовка реферата по одной из тем: – Технология сигналов и слотов – Характеристики объектно-ориентированных языков. Объекты. Классы – Программы с консольной графикой	**2**	**3**
	Содержание учебного материала: Проектирование диалоговых окон. Динамические диалоговые окна. Встроенные классы виджетов и диалоговых окон.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 3 Подготовка реферата по одной из тем: • Встроенные классы виджетов и диалоговых окон • Характеристики объектно-ориентированных языков. Наследование.	**2**	**3**

	Полиморфизм и перегрузка. • Отладка программ.		
	Лабораторная работа № 2 по теме «Разработка кода программного модуля с проектированием диалоговых окон. Выполнение отладки и тестирования программы на уровне модуля»	**2**	**2, 3**
	Лабораторная работа № 3 по теме «Выполнение отладки и тестирования программы на уровне модуля с использованием динамических диалоговых окон».	**2**	**2, 3**
Тема 1.3 Создание главных окон	**Содержание учебного материала:** Создание подкласса QMainWindow. Создание меню и панелей инструментов. Создание и настройка строки состояния. Реализация меню File. Применение диалоговых окон. Сохранение настроек приложения. Экранные заставки. Основные этапы разработки программного обеспечения.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 4 Подготовка реферата по одной из тем: • Директивы препроцессора. Заголовочные файлы. • Экранные заставки	**2**	**3**
	Лабораторная работа № 4 по теме «Выполнение отладки и тестирования программы на уровне модуля с реализацией меню File и применением диалоговых окон. Основные этапы разработки программного обеспечения. Сохранение настроек приложения»	**2**	**2, 3**
Тема 1.4 Реализация функциональности приложения	**Содержание учебного материала:** Основные принципы технологии структурного и объектно-ориентированного программирования. отладка и тестирование программы на уровне модуля. Центральный виджет.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 5. Подготовка реферата по одной из тем: • Подкласс QTableWidget. • Центральный виджет. Структуры. Перечисления.	**2**	**3**
	Содержание учебного материала: Создание подкласса QTableWidget. Загрузка и сохранение. Реализация меню Edit. Реализация других меню. Создание подкласса QTableWidget.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 6. Подготовка реферата по одной из тем: • Подкласс QTableWidget • Приоритеты операций языка C++.	**2**	**3**

	• Операторы языка C++.		
	Лабораторная работа № 5 по теме «Разработка кода программного модуля с созданием подкласса QTableWidget, загрузкой и сохранением. Отладка и тестирование программы на уровне модуля»	**2**	**2, 3**
	Лабораторная работа № 6 по теме «Разработка кода программного модуля с реализацией меню Edit. Реализация других меню. Создание подкласса QTableWidget. Отладка и тестирование программы на уровне модуля»	**2**	**2, 3**
Тема 1.5. Создание пользовательских виджетов	**Содержание учебного материала:** Основные принципы отладки и тестирования программных продуктов, использующих настройки виджетов Qt и создание подкласса QWidget. Интеграция пользовательских виджетов в QtDisigner. Двойная буферизация.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 7 Подготовка реферата по одной из тем: • Основные принципы отладки и тестирования программных продуктов. • Подкласс QWidget. Объекты и классы.	**2**	**3**
	Лабораторная работа № 7 по теме «Разработка кода программного модуля с настройкой виджетов Qt и созданием подкласса QWidget. Отладка и тестирование программы на уровне модуля»	**2**	**2, 3**
Тема 2. Средний уровень Qt программирования			
Тема 2.1 Управление компоновкой	**Содержание учебного материала:** Компоновка виджетов на форме. Стековая компоновка. Разделители.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 8 Подготовка реферата по одной из тем: • Стековая компоновка • Наследование. Базовые и производные классы.	**2**	**3**
	Содержание учебного материала: Области с прокруткой. Прикрепляемые виджеты и панели инструментов. Многодокументальный интерфейс.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 9 Подготовка реферата по одной из тем: • Прикрепляемые виджеты и панели инструментов • Указатели. Управление памятью.	**2**	**3**

	Лабораторная работа № 8 по теме «Основные принципы отладки и тестирования программного продукта, использующего компоновку виджетов на форме. Отладка и тестирование программы на уровне модуля»	**2**	**2, 3**
	Лабораторная работа № 9 по теме «Основные принципы отладки и тестирования программных продуктов, использующих многодокументальный интерфейс. Отладка и тестирование программы на уровне модуля»	**2**	**2, 3**
Тема 2.2 Обработка событий	**Содержание учебного материала:** Переопределение обработчиков событий. Установка фильтров событий. Обработка событий во время продолжительных процессов.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 10 Подготовка реферата по одной из тем: • Переопределение обработчиков событий • Виртуальные функции. Статические функции.	**2**	**3**
	Внеаудиторная самостоятельная работа № 11 Подготовка реферата по одной из тем: • Установка фильтров событий. • Потоковые классы.	**2**	**3**
	Лабораторная работа № 10 по теме «Установка фильтров событий».	**2**	**2,3**
	Лабораторная работа № 11 по теме «Обработка событий во время продолжительных процессов».	**2**	**2, 3**
Тема 2.3.Графика 2D и 3D.	**Содержание учебного материала:** Рисование при помощи QPainter. Преобразование рисовальщика.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 12 Подготовка реферата по одной из тем: • Рисование при помощи QPainter. Преобразование рисовальщика. • Создание многофайловых программ.	**2**	**3**
	Содержание учебного материала: Высококачественное воспроизведение изображения при помощи QImage. Вывод на печатающее устройство. Графические средства OpenGL.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 13 Подготовка реферата по одной из тем: • Вывод на печатающее устройство.	**2**	**3**

	• Шаблоны и исключения**.**		
	Лабораторная работа № 12 по теме «Разработка кода программного модуля с рисованием при помощи QPainter. Отладка и тестирование программы на уровне модуля»	**2**	**2, 3**
	Лабораторная работа № 13 по теме «Разработка кода программного модуля с рисованием при помощи графических средств OpenGL. Отладка и тестирование программы на уровне модуля»	**2**	**2,3**
	Лабораторная работа № 14 по теме «Вывод на печатающее устройство» .	**2**	
Тема 2.4 Технология “drag-and-drop”	**Содержание учебного материала:** Обеспечение поддержки технологии “drag-and-drop”.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 14 Подготовка реферата по одной из тем: • Технология “drag-and-drop”. • Стандартная библиотека шаблонов (STL).	**2**	**3**
	Содержание учебного материала: Поддержка пользовательских типов переносимых объектов. Работа с буфером обмена.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 15 Подготовка реферата по одной из тем: Буфер обмена. Разработка объектно-ориентированного ПО	**2**	**3**
	Лабораторная работа № 15 по теме «Разработка кода программного модуля с обеспечением поддержки технологии “drag-and-drop”.»	**2**	**2,3**
	Лабораторная работа № 16 по теме «Работа с буфером обмена»	**2**	**2,3**
Тема 2.5 Классы-контейнеры	**Содержание учебного материала:** Последовательные контейнеры. Ассоциативные контейнеры. Обобщённые алгоритмы. Строки, массивы байтов и объекты произвольного типа	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 16 Подготовка реферата по одной из тем: Обобщённые алгоритмы. Строки, массивы байтов и объекты произвольного типа.	**3**	**1, 2**

	Основы массивов. Массивы объектов. Массивы строк.		
	Лабораторная работа № 17 по теме «Разработка кода программного модуля с последовательными контейнерами. Ассоциативные контейнеры. . Обобщённые алгоритмы. Строки, массивы байтов и объекты произвольного типа»	**2**	**2, 3**
Тема 2.6 Ввод-вывод	**Содержание учебного материала:** Чтение и запись двоичных данных. Чтение и запись текста. Работа с каталогами. Ресурсы, внедрённые в исполняемый модуль. Связь между процессами.	**2**	**1, 2**
	Внеаудиторная самостоятельная работа № 17 Подготовка реферата по одной из тем: Связь между процессами Потоки и файлы. Потоковый ввод/вывод. Обработка ошибок ввода/вывода. Реагирование на ошибки. Анализ ошибок.	**2**	**3**
	Лабораторная работа № 18 по теме «Чтение и запись двоичных данных. Чтение и запись текста»	**2**	**2, 3**
	Лабораторная работа № 19 по теме «Ресурсы, внедрённые в исполняемый модуль». **Дифференцированный зачёт.**	**2**	**2, 3**
Итого по МДК 01.02. Прикладное программирование в т. ч.: теоретическое обучение - лабораторные занятия - внеаудиторная самостоятельная работа -		**105** **32** **38** **35**	
ПП.01.01. Производственная практика по профилю специальности к МДК 01.02. Прикладное программирование (6 семестр)		**108**	
Тема 1. Вводное занятие	***Виды работ*** Инструктаж о прохождении практики. Знакомство с программой практики и порядок её проведения, изучение правил внутреннего распорядка, знакомство с графиком работы студентов, ведения дневника практики, составление отчета. Инструктаж по технике безопасности, пожаробезопасности, производственной санитарии под роспись в журнале. Правила безопасности при работе с компьютером.	**6**	**3**
Тема 2. Разработка	***Виды работ*** Выбор задания. Анализ постановки задачи. Выбор методов и разработка основных	**6**	**3**

алгоритма поставленной задачи и реализация его средствами автоматизированного проектирования.	алгоритмов решения задачи. Разработка технического задания. Варианты заданий: – Создание диалоговых окон. – Создание главных окон. – Графика 2D и 3D. – Технология «drag-and-drop». – Управление компоновкой виджетов на форме. – Обработка событий во время продолжительных процессов. – Работа с каталогами.		
	Виды работ Разработка структуры и конкретных компонент разрабатываемого программного обеспечения, в том числе схемы алгоритмов, их общее описание.	**6**	**3**
	Виды работ Определение свойств входных и выходных данных поставленной задачи. Анализ процесса обработки информации и выбор структур данных для её хранения	**6**	**3**
Тема 3. Разработка кода программного продукта на основе готовой спецификации на уровне модуля.	***Виды работ*** Разработка структурной схемы программного продукта. Анализ и уточнение требований к программному продукту. Проектирование интерфейса пользователя. Создание главного окна. Добавление элементов управления (виджетов). Компоновка виджетов на форме. Использование технологии сигналов и слотов.	**6**	**3**
	Виды работ Создание меню, панелей инструментов и встроенных виджетов. Создание и настройка строки состояния, экранной заставки. Сохранение и загрузка настроек приложения	**6**	**3**
	Виды работ Проектирование отдельных классов. Создание и настройка модальных и немодальных диалоговых окон. Создание диалоговых окон сообщений. Разработка форм ввода-вывода данных. Чтение и запись текста. Работа с файлами.	**6**	**3**
	Виды работ Переопределение обработчиков событий. Установка фильтров событий. Обработка событий во время продолжительных процессов.	**6**	**3**
	Виды работ Разработка кода программного продукта согласно разработанному алгоритму в	**6**	**3**

	комплексной среде Qt.		
Тема 4. Использование инструментальных средств на этапе отладки программного продукта.	***Виды работ*** Отладка кода программного продукта, используя возможности отладчика комплексной среды Qt. Реализация диалога в графическом пользовательском интерфейсе.	**6**	**3**
	Виды работ Использование средств отладки, предоставляемых интерфейсом пользователя. Определение мест программы, в которых необходимо установить точки останова.	**6**	**3**
	Виды работ Использование инструмента отладки qDebug() и qWarning() для анализа значения переменных	**6**	**3**
Тема 5. Проведение тестирования программного модуля по определенному сценарию.	***Виды работ*** Структурное тестирование. Устранение утечки памяти. Исследование возможных причин утечки памяти.	**6**	**3**
	Виды работ Функциональное тестирование. Особенности отладки приложений, использующих шаблоны функций и классов.	**6**	**3**
	Виды работ Оценочное тестирование программного продукта. Анализ соответствия разработанного программного продукта постановке задачи.	**6**	**3**
Тема 6. Оформление документации на программные средства.	***Виды работ*** Составление программной документации. Определение сведений, необходимых для сопровождения и эксплуатации программного продукта. Разработка пояснительной записки, содержащей информацию о структуре и конкретных компонентах программного обеспечения, в том числе схемы алгоритмов, их общее описание, обоснование принятых технических решений	**6**	**3**
	Виды работ Разработка описания применения, содержащего сведения о назначении программного продукта, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств.	**6**	**3**
	Виды работ Разработка руководства системного программиста, содержащего сведения для проверки, обеспечения функционирования и настройки программы на условия	**6**	**3**

	конкретного применения		
Итого по ПМ.01 Разработка программных модулей программного обеспечения компьютерных систем		**456**	
в том числе:			
теоретическое обучение		**160**	
лабораторные работы		**84**	
практические работы		**-**	
курсовая работа		**-**	
внеаудиторная самостоятельная работа студента		**80**	
учебная практика		**108**	
производственная практика (по профилю специальности)		**108**	

4. УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

4.1. Требования к минимальному материально-техническому обеспечению

Реализация программы **ПМ.01 Разработка программных модулей программного обеспечения компьютерных систем** предполагает наличие учебных лабораторий: **системного и прикладного программирования, информационно-коммуникационных систем.**

1. Лаборатория системного и прикладного программирования, оснащённая оборудованием:

– посадочные места по количеству обучающихся (столы, стулья по числу посадочных мест);

– рабочее место преподавателя (стол, стул);

– учебно-методический комплекс по дисциплине (рабочие программы, календарно-тематические планы, разработки уроков по дисциплине, учебно-методическое обеспечение к каждому уроку, в т.ч. презентации к урокам, комплект видеоуроков, комплект контрольно-оценочных средств и др.);

– программное обеспечение общего назначения.

– локальная сеть.

с техническими средствами обучения:

– персональные компьютеры с лицензионным программным обеспечением, мультимедийная система.

Программное обеспечение:

– операционная система Windows XP, Vista,7;

– средаVisual Studio 2010 Release Candidate илиVisual Studio 2005/2008;

– язык программирования Microsoft Visual C++

– средств разработки Qt by Digiav4.8.5. (MinGW OpenSource);

– пакет прикладных программ Microsoft Office.

2. Лаборатория информационно-коммуникационных технологий, оснащённая оборудованием:

– посадочные места по количеству обучающихся (столы, стулья по числу посадочных мест);

– рабочее место преподавателя (стол, стул);

– комплект учебно-методической документации;

– программное обеспечение общего назначения.

– локальная сеть.

с техническими средствами обучения:

– автоматизированное рабочее место преподавателя;

– автоматизированные рабочие места обучающихся;

– компьютер с выходом в Интернет;

– интерактивная доска;

– компьютеры;

– программное обеспечение общего и профессионального назначения.

4.2 Информационное обеспечение реализации программы

Для реализации программы библиотечный фонд илиала имеет печатные и /или электронные образовательные ресурсы, рекомендуемые для использования в образовательном процессе

4.2.1 Основная литература:

1. Брайен, Керниган В. Язык программирования С : учебник / Брайен Керниган В., Деннис Ричи М. — Москва : Интуит НОУ, 2016. — 313 с. — URL: https://book.ru/book/918294
2. Александров, Э.Э. Программирование на языке С в Microsoft Visual Studio 2010 : курс лекций / Александров Э.Э., Афонин В.В. — Москва : Интуит НОУ, 2016. — 570 с. — URL: https://book.ru/book/918122
3. Алексеев, Е.Р. Программирование на языке C++ в среде Qt Creator : курс лекций / Алексеев Е.Р., Злобин Г.Г., Костюк Д.А., Чеснокова О.В., Чмыхало А.С. — Москва : Интуит НОУ, 2016. — 715 с. — URL: https://book.ru/book/918128

4.2.2 Электронные ресурсы:

1. ЭБС ИЗДАТЕЛЬСТВА "BOOK.RU" КОЛЛЕКЦИЯ СПО https://www.book.ru/
2. ЭБС ИЗДАТЕЛЬСТВА "ЮРАЙТ" https://urait.ru
3. ЭБС ИЗДАТЕЛЬСТВА "ЛАНЬ" https://e.lanbook.com

4.3 Дополнительная литература:

1. Макарова, Н.В. Основы программирования : учебник / Макарова Н.В., Нилова Ю.Н., Зеленина С.Б., Лебедева Е.В. — Москва : КноРус, 2021. — 451 с. — ISBN 978-5-406-03394-4. — URL: https://book.ru/book/936582
2. Фридман, А.Л. Язык программирования С : курс лекций / Фридман А.Л. — Москва : Интуит НОУ, 2016. — 218 с. — ISBN 978-5-9556-0017-8. — URL: https://book.ru/book/918295
3. Златопольский, Д.М. Программирование: типовые задачи, алгоритмы, методы : учебное пособие / Златопольский Д.М. 4-е изд. — Москва : Лаборатория знаний, 2020. — 224 с. — ISBN 978-5-00101-789-9. — URL: https://book.ru/book/936428

4.4 Общие требования к организации образовательного процесса

Освоение обучающимися профессионального модуля проходит в условиях созданной образовательной среды как в учебном заведении, так и в организациях, соответствующих профилю специальности изучаемого модуля**.**

Освоение программы модуля **ПМ.01 Разработка программных модулей программного обеспечения компьютерных систем** заключается в изучении междисциплинарных курсов **МДК 01.01. Системное программирование и МДК 01.02. Прикладное программирование** Реализация программы предполагает учебную практику после изучения междисциплинарного курса МДК.01.01 и производственную практику по профилю специальности - после изучения МДК.01.02.

Освоению профессионального модуля предшествует изучение профильной дисциплины: «Информатика» и общепрофессиональных дисциплин: «Операционные системы», «Архитектура компьютерных систем», «Основы программирования», «Теория алгоритмов» и др..

Изучение программы каждого междисциплинарного курса завершается дифференцированным зачетом студентов по освоенным общим и профессиональным компетенциям, указанным в данном курсе.

Учебная практика и практика производственная (по профилю специальности) завершаются дифференцированным зачетом студентов по освоенным общим и профессиональным компетенциям.

Изучение программы модуля завершается промежуточной аттестацией в форме квалификационного экзамена, результаты которого оцениваются на основании выполнения студентами всех зачетных мероприятий по модулю.

4.5. Кадровое обеспечение образовательного процесса

Требования к квалификации педагогических (инженерно-педагогических) кадров, обеспечивающих обучение обучающихся по междисциплинарному курсу:

– наличие высшего профессионального образования, соответствующего профилю профессионального модуля;

– наличие опыта практической деятельности по профилю направления профессионального модуля;

– стажировка на профильном предприятии не реже 1 раза в 3 года.

Для мастера производственного обучения обязательным является стажировка в профильных организациях не реже 1-го раза в 3 года; опыт деятельности в организациях соответствующей профессиональной сферы.

Данные о педагогическом работнике, осуществляющем реализацию профессионального модуля, представлены в таблице 4.

Таблица 4 - Характеристика педагогического работника, обеспечивающего учебный процесс по профессиональному модулю

№ п/п	Наименование предмета, дисциплины, модуля, профессионального модуля в соответствии с учебным планом	Характеристика педагогических работников					
		Фамилия, имя, отчество, должность по штатному расписанию	Какое образовательное учреждение окончил, специальность (направление подготовки) по документу об образовании	Ученая степень и ученое (почетное) звание, квалификационная категория	Стаж педагогической (научно-педагогической) работы	Основное место работы, должность	Условия привлечения к педагогической деятельности
1	2	3	4	5	6	7	8
	МДК 01.01. Системное программирование	Е.В. Волошин	ФГБОУ ВО «ВГУЭС» в г. Артеме, бакалавр по направлению 09.03.03 Прикладная информатика	Преподаватель	2 г.9 мес.	преподаватель филиала ФГБОУ ВО «ВГУЭС» в г. Артеме	штатный
	МДК 01.02. Прикладное программирование	Е.В. Волошин	ФГБОУ ВО «ВГУЭС» в г. Артеме, бакалавр по направлению 09.03.03 Прикладная информатика	Преподаватель	2 г.9 мес.	преподаватель филиала ФГБОУ ВО «ВГУЭС» в г. Артеме	штатный
	УП. 01.01 Учебная практика	Е.В. Волошин	ФГБОУ ВО «ВГУЭС» в г. Артеме, бакалавр по направлению 09.03.03 Прикладная информатика	Преподаватель	2 г.9 мес.	преподаватель филиала ФГБОУ ВО «ВГУЭС» в г. Артеме	штатный
	ПП. 01.01 Производственная практика по профилю специальности	Е.В. Волошин	ФГБОУ ВО «ВГУЭС» в г. Артеме, бакалавр по направлению 09.03.03 Прикладная информатика	Преподаватель	2 г.9 мес.	преподаватель филиала ФГБОУ ВО «ВГУЭС» в г. Артеме	штатный

5. КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ ПРОФЕССИОНАЛЬНОГО МОДУЛЯ ПМ.01 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНЫХ СИСТЕМ

5.1 Контроль и оценка сформированности профессиональных компетенций обучающихся

Таблица 5. – Контроль и оценка сформированности профессиональных компетенций обучающихся

Результаты (освоенные профессиональные компетенции)	**Основные показатели оценки результата**	**Формы и методы контроля и оценки**
ПК 1.1. Выполнять разработку спецификаций отдельных компонент.	Точность определения основных этапов разработки программного обеспечения. Правильность применения основных принципов технологии структурного и объектно-ориентированного программирования. Правильность оформления документации на программные средства. Правильность и точность разработки алгоритма поставленной задачи.	Текущий контроль в форме: - защиты лабораторных заданий; - контрольных работ по темам МДК. Зачеты по производственной практике и по каждому из разделов профессионального модуля.
ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.	Правильность применения основных принципов технологии структурного и объектно-ориентированного программирования. Правильность и точность разработки кода программного модуля на современных языках программирования. Точность создания программы по разработанному алгоритму как отдельного модуля. Правильность разработки кода программного продукта на основе готовой спецификации на уровне модуля.	Текущий контроль в форме: - защиты лабораторных и практических занятий; - контрольных работ по темам МДК. Зачеты по каждому из разделов междисциплинарного курса.
ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.	Правильность применения основных принципов отладки и тестирования программных продуктов. Точность использования инструментальных средств на этапе	Текущий контроль в форме: - защиты лабораторных и практических занятий; - контрольных работ по темам МДК. Зачеты по производственной

	отладки программного продукта. Правильность отладки и тестирования программы на уровне модуля.	практике и по каждому из разделов МДК.
ПК 1.4. Выполнять тестирование программных модулей.	Проведение тестирования программного модуля по разработанному сценарию. Правильность выполнения отладки и тестирование программы на уровне модуля.	Текущий контроль в форме: - защиты лабораторных и практических занятий; - контрольных работ по темам МДК. .
ПК 1.5. Осуществлять оптимизацию программного кода модуля.	Точность проведения оптимизации программного кода модуля по определенному сценарию. Правильность выполнения отладки и тестирование программы на уровне модуля. Правильность использования инструментальных средств на этапе отладки программного продукта.	Текущий контроль в форме: - защиты лабораторных занятий; - контрольных работ по темам МДК.
ПК 1.6. Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.	Правильность использования инструментальных средств для автоматизации оформления документации. Правильность определения и использования методов и средств разработки технической документации.	Текущий контроль в форме: - защиты лабораторных занятий; - контрольных работ по темам МДК. .

Формы и методы контроля и оценки результатов обучения позволяют проверять у обучающихся не только сформированность профессиональных компетенций, но и развитие общих компетенций и обеспечивающих их умений.

5.2. Контроль и оценка результатов развития общих компетенций и обеспечивающих их умений

Таблица 6. – Контроль и оценка результатов развития общих компетенций и обеспечивающих их умений (базовая подготовка)

Результаты (освоенные общие компетенции)	**Основные показатели результатов подготовки**	**Формы и методы контроля**
ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес	Демонстрация устойчивого интереса к будущей профессии	Экспертное наблюдение и оценка деятельности обучающегося в процессе освоения образовательной программы на практических занятиях, при выполнении работ по учебной и производственной практике. Экспертное наблюдение и оценка активности учащегося при проведении учебно-воспитательных мероприятий профессиональной направленности.
ОК 2. Организовывать собственную деятельность, выбирать типо-	Мотивированное обоснование выбора и применения методов и способов	Экспертное наблюдение и оценка деятельности обучающегося в процессе освоения образовательной программы на

Результаты (освоенные общие компетенции)	Основные показатели результатов подготовки	Формы и методы контроля
вые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество	решения профессиональных задач при осуществлении выполнения задания. Точность, правильность и полнота выполнения профессиональных задач	практических занятиях, при выполнении работ по учебной и производственной практике.
ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность	Демонстрация способности принимать оперативные решения в стандартных и нестандартных ситуациях, нести за них ответственность при выполнении задания.	Наблюдение и оценка результатов деятельности обучающегося в процессе освоения образовательной программы на практических занятиях, при выполнении индивидуальных домашних заданий. Наблюдение и оценка активности учащихся при проведении учебно-воспитательных мероприятий профессиональной направленности.
ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития	Оперативность поиска и использования необходимой информации для качественного выполнения профессиональных заданий, профессионального и личностного развития	Наблюдение и оценка деятельности обучающегося в процессе освоения образовательной программы на практических занятиях, при выполнении индивидуальных домашних заданий, работ по учебной и производственной практике
ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности	Демонстрация умения оперативно осуществлять операции, предлагаемые преподавателем, делать анализ и давать оценку полученной информации, в т.ч. и с использованием программного обеспечения	Экспертное наблюдение и оценка деятельности обучающегося в процессе освоения образовательной программы на практических занятиях, в ходе компьютерного тестирования, подготовки электронных презентаций, при выполнении индивидуальных домашних заданий. Наблюдение и оценка использования учащихся информационных технологий при подготовке и проведении учебно-воспитательных мероприятий различной тематики.
ОК 6. Работать в коллективе и команде, эффективно общаться с коллегами, руководством, потребителями	Коммуникабельность при взаимодействии с обучающимися и преподавателями в ходе обучения. Воспитание уважения к мнению сокурсников.	Экспертное наблюдение и оценка коммуникативной деятельности обучающегося в процессе освоения образовательной программы на практических занятиях, при выполнении индивидуальных домашних заданий. Наблюдение и оценка использования учащихся коммуникативных методов и приемов при подготовке и проведении учебно-воспитательных мероприятий различной тематики.
ОК 7. Брать на себя ответственность за рабо-	Ответственность за результат выполнения	Экспертное наблюдение и оценка деятельности обучающегося в процессе

Результаты (освоенные общие компетенции)	Основные показатели результатов подготовки	Формы и методы контроля
ту членов команды (подчиненных), результат выполнения заданий	задания. Способность к самоанализу и коррекции результатов собственной работы.	освоения образовательной программы на практических занятиях при работе в малых группах. Экспертное наблюдение и оценка уровня ответственности учащегося при подготовке и проведении учебно-воспитательных мероприятий различной тематики. Экспертное наблюдение и оценка динамики достижений учащегося в учебной и общественной деятельности.
ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации	Способность к организации и планированию самостоятельных занятий при изучении дисциплины. Демонстрация потребности в получении дополнительных знаний, возможностей самореализации.	Экспертное наблюдение и оценка использования учащимися методов и приемов личной организации в процессе освоения образовательной программы на практических занятиях, при выполнении индивидуальных домашних заданий. Экспертное наблюдение и оценка использования учащимися методов и приемов личной организации при подготовке и проведении учебно-воспитательных мероприятий различной тематики. Экспертное наблюдение и оценка динамики достижений учащихся в учебной и общественной деятельности.
ОК 9. Ориентироваться в условиях частой смены технологий в профессиональной деятельности	Проявление интереса к инновациям в области профессиональной деятельности	Экспертное наблюдение и оценка деятельности учащегося в процессе освоения образовательной программы на практических занятиях, при выполнении индивидуальных домашних заданий.

5.3. Оценка индивидуальных образовательных достижений по результатам текущего контроля и промежуточной аттестации

Таблица 8. – Оценка индивидуальных образовательных достижений по результатам текущего контроля и промежуточной аттестации

№ п/п	Баллы по рейтингу	Качественная оценка индивидуальных образовательных достижений	
		вербальный аналог	балл (отметка)
1	41	допуск к аттестации	
2	менее 61	неудовлетворительно	2
3	61-75	удовлетворительно	3
4	76-90	хорошо	4
5	91-100	отлично	5
6.	61-100	зачтено	
7.	менее 61	не зачтено	

5.4 Критерии оценки знаний студентов по междисциплинарному курсу

МДК.01.01. Системное программирование (5 семестр)

п/п	Наименование работ	Всего баллов 100			
		Текущая аттестация от 0 до 40 баллов (1-8 неделя)		Семестровая аттестация от 60 до 100 баллов (9-16 неделя)	
		знания, умения	компетенции	знания, умения	компетенции
1	Наличие теоретического материала	2,5	2,5	2,5	2,5
2	Посещаемость	5	-	5	-
3	Практические занятия	7,5	7,5	7,5	7,5
4	Индивидуальные домашние задания	7,5	7,5	7,5	7,5
5	Экзамен	-	-	10	10
Итого:		**40**		**60**	

МДК.01.02. Прикладное программирование (6 семестр)

№ п/п	Наименование работ	Всего баллов 100			
		Текущая аттестация от 0 до 40 баллов (1-8 неделя)		Семестровая аттестация от 60 до 100 баллов (9-16 неделя)	
		знания, умения	компетенции	знания, умения	компетенции
1	Наличие теоретического материала	2,5	2,5	2,5	2,5
2	Посещаемость	5	-	5	-
3	Практические занятия	7,5	7,5	7,5	7,5
4	Индивидуальные домашние задания	7,5	7,5	7,5	7,5
5	Экзамен	-	-	10	10
Итого:		**40**		**60**	

На этапе промежуточной аттестации по медиане качественных оценок индивидуальных образовательных достижений экзаменационной комиссией определяется интегральная оценка освоенных обучающимися профессиональных и общих компетенций как результатов освоения профессионального модуля.

Приложение А

Таблица А. –ГЛОССАРИЙ ОСНОВНЫХ ТЕРМИНОВ И ОПРЕДЕЛЕНИЙ, ИЗУЧАЕМЫХ В МОДУЛЕ

Термин	Определение
Алгоритм	формально описанная последовательность действий, которые необходимо выполнить для получения требуемого результата
Виджет	(widget) любой визуальный элемент графического интерфейса пользователя (кнопки меню, полосы прокрутки, фреймы - примеры виджетов).
Выражение	Все вычисления в программе записываются в виде выражений
Диалог	регламентированный обмен информацией между человеком и компьютером
Запись	Структура данных, состоящая из фиксированного числа компонент
Консоль	в различных программах и играх **консолью** стали называть **окно** для вывода системных сообщений и приёма команд.
Меню	Выводится на экран при записи программы
Модуль	Автономно компилируемая коллекция программных ресурсов
Библиотека Win32 API	библиотека программного интерфейса приложений
Классы	экземпляры определённого типа свойств, образующие иерархию с наследованием свойств.
Объекты	элементы конкретной предметной области.
Объектно-ориентированное программирование	технология создания сложного программного обеспечения, основанная на представлении программы в виде совокупности объектов, каждый из которых является классом.
Отладка	процесс локализации и исправления ошибок, обнаруженных при тестировании программного продукта.
Сообщение	порция информации, участвующая в диалоговом обмене.
Среда программирования	программный комплекс, который включает специализированный текстовый редактор, встроенные компилятор, компоновщик, отладчик, справочную систему.
Проектирование	Детальная проработка последовательности действий будующей программы

Приложение Б.

Таблица Б. –ЛИСТ ИЗМЕНЕНИЙИ ДОПОЛНЕНИЙ, ВНЕСЁННЫХ В РАБОЧУЮ ПРОГРАММУ

№ изменения, дата внесения изменения, № страницы с изменением:	
БЫЛО:	**СТАЛО:**
Основание: Подпись лица, внесшего изменения	

№ изменения, дата внесения изменения, № страницы с изменением:	
БЫЛО:	**СТАЛО:**
Основание: Подпись лица, внесшего изменения	

ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ «ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ЭКОНОМИКИ И СЕРВИСА» В Г. АРТЕМЕ

Техническая экспертиза программы профессионального модуля ПМ.01 Разработка программных модулей программного обеспечения компьютерных систем по специальности 09.02.03 Программирование в компьютерных системах, представленного преподавателем кафедры экономики, управления и информационных технологий филиала Волошиным Е.В.

ЭКСПЕРТНОЕ ЗАКЛЮЧЕНИЕ

№	Наименование экспертного показателя	Экспертная оценка	
		да	нет
Экспертиза оформления титульного листа и содержания			
1.	Наименование рабочей программы модуля на титульном листе совпадает с наименованием модуля в тексте ФГОС и учебном плане колледжа	да	
2.	Название филиала соответствует названию по Уставу	да	
3.	На титульном листе указан код и наименование специальности	да	
4.	Оборотная сторона титульного листа содержит все необходимые данные (выходные данные ФГОС, ФИО экспертов, сведения о согласовании программы)	да	
5.	Нумерация страниц в «Содержании» верна	да	
Экспертиза раздела 1 «Паспорт рабочей программы профессионального модуля»			
6.	Раздел 1 «Паспорт рабочей программы профессионального модуля» имеется	да	
7.	Подстрочные надписи удалены	да	
8.	Наименование программы профессионального модуля совпадает с наименованием на титульном листе	да	
9.	Пункт 1.1. «Область применения программы» заполнен	да	
10.	Паспорт программы профессионального модуля содержит базовую часть	да	
11.	Наименование основного вида профессиональной деятельности (ВПД) совпадает с наименованием профессионального модуля	да	
12.	Перечень профессиональных компетенций (ПК) содержит все компетенции, перечисленные в тексте ФГОС	да	
13	Пункт 1.2. «Цели и задачи модуля – требования к результатам освоения модуля» заполнен	да	
14.	Требования к практическому опыту, умениям и знаниям соответствуют перечисленным в тексте ФГОС	да	
15.	Пункт 1.3. «Рекомендуемое количество часов на освоение программы профессионального модуля» заполнен	да	
16.	Все строки и графы пункта 1.3. заполнены	да	

17.	Перечислены виды самостоятельной работы	**да**	
18.	Указанное количество часов в графе «Итого» соответствует учебному плану	**да**	
Экспертиза раздела 2 «Результаты освоения профессионального модуля»			
19.	Раздел 2 «Результаты освоения профессионального модуля» имеется	**да**	
20.	Перечень профессиональных компетенций совпадает с указанными в п. 1.1	**да**	
21.	Перечень общих компетенций соответствует перечисленным в тексте ФГОС	**да**	
Экспертиза раздела 3 «Структура и содержание профессионального модуля»			
22.	Раздел 3 «Структура и содержание профессионального модуля» имеется	**да**	
23.	Форма таблицы 3.1. «Тематический план профессионального модуля» соответствует макету программы	**да**	
24.	Таблица 3.1. «Тематический план профессионального модуля» заполнена	**да**	
25.	Форма таблицы 3.2. «Содержание обучения по профессиональному модулю (ПМ)» соответствует макету программы	**да**	
26.	Таблица 3.2. «Содержание обучения по профессиональному модулю (ПМ)» заполнена	**да**	
27.	Количество и наименования междисциплинарных курсов совпадают с указанными в тексте ФГОС	**да**	
28.	Перечислены виды самостоятельной работы студентов, сформулированные через деятельность	**да**	
29.	Наименования разделов модуля в табл. 3.1 и 3.2 совпадают	**да**	
Экспертиза раздела 4 «Условия реализации программы профессионального модуля»			
30.	Раздел 4 «Условия реализации программы профессионального модуля» имеется	**да**	
31.	Пункт 4.1. «Требования к минимальному материально-техническому обеспечению» заполнен	**да**	
32.	Пункт 4.2. «Информационное обеспечение обучения» заполнен в соответствии с требованиями ГОСТ по оформлению литературы	**да**	
33.	В списке основной литературы отсутствуют издания, выпущенные более 5 лет назад	**да**	
34.	Пункт 4.3. «Общие требования к организации образовательного процесса» заполнен	**да**	
35.	Пункт 4.4. «Кадровое обеспечение образовательного процесса» заполнен	**да**	
Экспертиза раздела 5 «Контроль и оценка результатов освоения профессионального модуля(вида профессиональной деятельности)»			
36.	Раздел 5. «Контроль и оценка результатов освоения профессионального модуля (вида профессиональной деятельности)» имеется	**да**	
37.	Наименования профессиональных и общих компетенций совпадают с указанными в п. 1.1	**да**	
Экспертиза показателей объемов времени, отведенных на освоение ПМ, указанных в п. 1.3 раздела 1 «Паспорт рабочей программы профессионального модуля» и в табл. 3.1 и 3.2 раздела 3 «Структура и содержание профессионального модуля»			
38.	Общий объем времени, отведенного на освоение модуля (всего часов), в паспорте программы, таблицах 3.1 и 3.2 совпадает	**да**	
39.	Объем обязательной аудиторной нагрузки в паспорте программы, таблицах 3.1 и 3.2 совпадает	**да**	
40.	Объем времени, отведенного на выполнение лабораторных и практических занятий, в таблицах	**Да**	

	3.1 и 3.2 совпадает		
41.	Объем времени, отведенного на практику, в паспорте программы, таблицах 3.1 и 3.2 совпадает	**Да**	
42.	Объем времени, отведенного на самостоятельную работу студентов, в паспорте программы, таблицах 3.1 и 3.2 совпадает	**Да**	

ИТОГОВОЕ ЗАКЛЮЧЕНИЕ		**Да**	**нет**
Программа профессионального модуля может быть направлена на содержательную экспертизу		**Да**	

Разработчик программы: _____ Е.В. Волошин

«22» апреля 2020 г.

СОГЛАСОВАНО

Заведующий кафедрой _____ А.А.Власенко

Заведующий отделением _____ М.С.Словикова

Методист УМЧ _____ Т.И. Теплякова

«25 » апреля 2020 г.

ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ЭКОНОМИКИ И СЕРВИСА» В Г. АРТЕМЕ

Содержательная экспертиза рабочей программы профессионального модуля ПМ.01 Разработка программных модулей программного обеспечения компьютерных систем по специальности 09.02.03 Программирование в компьютерных системах, представленного преподавателем кафедры экономики, управления и информационных технологий филиала Волошиным Е.В.

ЭКСПЕРТНОЕ ЗАКЛЮЧЕНИЕ

№	Наименование экспертного показателя	Экспертная оценка			Примечание
		да	нет	заключение отсутствует	
Экспертиза раздела 1 «Паспорт программы профессионального модуля»					
1.	Требования к результатам освоения модуля соответствуют перечисленным в ФГОС СПО (в т. ч. конкретизируют и/или расширяют требования ФГОС)	**да**			
2.	Возможности использования профессионального модуля описаны полно и точно.	**да**			
3.	Вариативная часть содержит требования к результатам освоения модуля(при наличии)	**да**			
Экспертиза раздела 2 «Результаты освоения профессионального модуля»					
4.	Перечень общих и профессиональных компетенций, представленных в разделе модуля, соответствует перечисленным компетенциям, указанным к данному модулю в ФГОС СПО.	**да**			
Экспертиза раздела 3 «Структура и содержание профессионального модуля»					
5.	Структура программы профессионального модуля соответствует принципу единства теоретического и практического обучения.	**да**			
6	Разделы программы модуля выделены дидактически целесообразно.	**да**			
7.	Соотношение учебной и производственной практики соответствует дидактическим задачам модуля.	**да**			
8.	Выбор варианта проведения практики (концентрированно, рассредоточено, комбинированно) дидактически целесообразен.	**да**			
9.	Содержание практики (виды работ) соответствует требованиям к практическому опыту и умениям	**да**			
10.	Содержание учебного материала соответствует требованиям к знаниям и умениям.	**да**			
11.	Объем времени достаточен для освоения указанного в содержании учебного материала.	**да**			
12.	Объем и содержание лабораторных и практических работ определены дидактически целесообразно и соответствуют требованиям к умениям и знаниям и ориентированы на подготовку к овладению ПК профессионального модуля.	**да**			
13.	Уровни освоения соответствуют видам учебной деятельности в разделе.	**да**			
14.	Тематика домашних заданий определена дидактически целесообразно.	**да**			
15.	Содержание самостоятельной работы студентов, в т.ч. внеаудиторной, направлено на выполнение требований к результатам освоения ПМ («иметь практический	**да**			

	опыт», «уметь», «знать»).				
16.	Формулировки самостоятельной работы понимаются однозначно.	**да**			
17.	Примерная тематика курсовых работ соответствует целям и задачам освоения профессионального модуля *(пункт заполняется, если в программе дисциплины предусмотрена курсовая работа)если в программе дисциплины предусмотрена курсовая работа)*	**не предусмотрена**			
18.	Содержание программы модуля предусматривает формирование перечисленных общих и профессиональных компетенций.	**да**			
Экспертиза раздела 4 «Условия реализации программы профессионального модуля»					
19.	Перечень учебных кабинетов (мастерских, лабораторий и др.) обеспечивает изучение междисциплинарного курса, проведение всех видов лабораторных и практических работ и тем учебной практики, предусмотренных программой профессионального модуля.	**да**			
20.	Перечисленное оборудование обеспечивает изучение междисциплинарного курса, проведение всех видов лабораторных и практических работ и тем учебной практики, предусмотренных программой профессионального модуля.	**да**			
21.	Перечень рекомендуемой основной и дополнительной литературы включает общедоступные источники.	**да**			
22.	Перечисленные Интернет-ресурсы актуальны и достоверны (пункт заполняется, если нормативно-правовые акты указаны в качестве источников).	**да**			
23.	Перечисленные источники соответствуют структуре и содержанию программы модуля.	**да**			
24.	Информационные источники указаны с учетом содержания модуля.	**да**			
25.	Общие требования к организации образовательного процесса соответствуют модульно - компетентностному подходу.	**да**			
26.	Общие требования к организации образовательного процесса описаны подробно (перечислены условия проведения занятий, организация учебной практики, консультационной помощи обучающимся).	**да**			
27.	Дисциплины, изучение которых должно предшествовать освоению данного модуля, определены с учетом принципа систематичности и последовательности обучения.	**да**			
28.	Требования к кадровому обеспечению (в т.ч. к уровню квалификации преподавателей МДК и руководителя практики) позволяют обеспечить должный уровень подготовки современного рабочего (специалиста).	**да**			
Экспертиза раздела 5 «Контроль и оценка результатов освоения профессионального модуля (вида профессиональной деятельности)»					
29.	Основные показатели оценки результатов обучения позволяют однозначно диагностировать сформированность соответствующих профессиональных компетенций (ПК).	**да**			
	Наименование форм и методов контроля и оценки освоения ПК точно и однозначно описывает процедуру аттестации.	**да**			
30.	Формы и методы контроля и оценки освоения ПК позволяют оценить сформированность ПК.	**да**			

31.	Основные показатели оценки результата позволяют однозначно диагностировать сформированность соответствующих общих компетенций (ОК).	**да**			
32.	Наименование форм и методов контроля и оценки освоения ОК точно и однозначно описывает процедуру аттестации.	**да**			
33.	Формы и методы контроля и оценки освоения ОК позволяют оценить сформированность ОК.	**да**			

Итоговое заключение (из трех альтернативных позиций следует выбрать одну)	**да**	**нет**
Программа профессионального модуля может быть рекомендована к утверждению	да	
Программу профессионального модуля следует рекомендовать к доработке		
Программу профессионального модуля следует рекомендовать к отклонению		

Замечания и рекомендации эксперта по доработке_____

Разработчик программы: _____ Е.В. Волошин

СОГЛАСОВАНО

И.о.Заведующий кафедрой _____ А.А.Власенко

«22» апреля 2020 г.

Заведующий отделением _____ М.С.Словикова

Методист УМЧ _____ Т.И. Теплякова

«25 » апреля 2020 г.

ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ЭКОНОМИКИ И СЕРВИСА» В Г. АРТЕМЕ
Содержательная экспертиза рабочей программы профессионального модуля
ПМ.01 Разработка программных модулей программного обеспечения компьютерных систем по специальности 09.02.03 Программирование в компьютерных системах, представленного преподавателем кафедры экономики, управления и информационных технологий филиала Волошиным Е.В.
ЭКСПЕРТНОЕ ЗАКЛЮЧЕНИЕ

№	Наименование экспертного показателя	Экспертная оценка			Примечание
		да	нет	заключение отсутствует	
Экспертиза раздела 1 «Паспорт программы профессионального модуля»					
1.	Требования к результатам освоения модуля соответствуют перечисленным в ФГОС СПО (в т. ч. конкретизируют и/или расширяют требования ФГОС)	**да**			
2.	Возможности использования профессионального модуля описаны полно и точно.	**да**			
3.	Вариативная часть содержит требования к результатам освоения дисциплины (при наличии)	**да**			
Экспертиза раздела 2 «Результаты освоения профессионального модуля»					
4.	Перечень общих и профессиональных компетенций, представленных в разделе модуля, соответствует перечисленным компетенциям, указанным к данному модулю в ФГОС СПО.	**да**			
Экспертиза раздела 3 «Структура и содержание профессионального модуля»					
5.	Структура программы профессионального модуля соответствует принципу единства теоретического и практического обучения.	**да**			
6	Разделы программы модуля выделены дидактически целесообразно.	**да**			
7.	Соотношение учебной и производственной практики соответствует дидактическим задачам модуля.	**да**			
8.	Выбор варианта проведения практики (концентрированно, рассредоточено, комбинированно) дидактически целесообразен.	**да**			
9.	Содержание практики (виды работ) соответствует требованиям к практическому опыту и умениям	**да**			
10.	Содержание учебного материала соответствует требованиям к знаниям и умениям.	**да**			
11.	Объем времени достаточен для освоения указанного в содержании учебного материала.	**да**			
12.	Объем и содержание лабораторных и практических работ определены дидактически целесообразно и соответствуют требованиям к умениям и знаниям и ориентированы на подготовку к овладению ПК профессионального модуля.	**да**			

13.	Уровни освоения соответствуют видам учебной деятельности в разделе.	**да**			
14.	Тематика домашних заданий определена дидактически целесообразно.	**да**			
15.	Содержание самостоятельной работы студентов, в т.ч. внеаудиторной, направлено на выполнение требований к результатам освоения ПМ («иметь практический опыт», «уметь», «знать»).	**да**			
16.	Формулировки самостоятельной работы понимаются однозначно.	**да**			
17.	Примерная тематика курсовых работ соответствует целям и задачам освоения профессионального модуля *(пункт заполняется, если в программе дисциплины предусмотрена курсовая работа)если в программе дисциплины предусмотрена курсовая работа)*	**не предусмотрена**			
18.	Содержание программы модуля предусматривает формирование перечисленных общих и профессиональных компетенций.	**да**			
Экспертиза раздела 4 «Условия реализации программы профессионального модуля»					
19.	Перечень учебных кабинетов (мастерских, лабораторий и др.) обеспечивает изучение междисциплинарных курсов, проведение всех видов лабораторных и практических работ и тем учебной практики, предусмотренных программой профессионального модуля.	**да**			
20.	Перечисленное оборудование обеспечивает изучение междисциплинарных курсов, проведение всех видов лабораторных и практических работ и тем учебной практики, предусмотренных программой профессионального модуля.	**да**			
21.	Перечень рекомендуемой основной и дополнительной литературы включает общедоступные источники.	**да**			
22.	Перечисленные Интернет-ресурсы актуальны и достоверны (пункт заполняется, если нормативно-правовые акты указаны в качестве источников).	**да**			
23.	Перечисленные источники соответствуют структуре и содержанию программы модуля.	**да**			
24.	Информационные источники указаны с учетом содержания модуля.	**да**			
25.	Общие требования к организации образовательного процесса соответствуют модульно - компетентностному подходу.	**да**			
26.	Общие требования к организации образовательного процесса описаны подробно (перечислены условия проведения занятий, организация учебной практики , консультационной помощи обучающимся).	**да**			
27.	Дисциплины, изучение которых должно предшествовать освоению данного модуля, определены с учетом принципа систематичности и последовательности обучения.	**да**			
28.	Требования к кадровому обеспечению (в т.ч. к уровню квалификации преподавателей МДК и руководителя практики) позволяют обеспечить должный уровень подготовки современного рабочего (специалиста).	**да**			
Экспертиза раздела 5 «Контроль и оценка результатов освоения профессионального модуля (вида профессиональной деятельности)»					
29.	Основные показатели оценки результатов обучения позволяют однозначно диагностировать сформированностьсоответствующих профессиональных компетенций (ПК).	**да**			

	Наименование форм и методов контроля и оценки освоения ПК точно и однозначно описывает процедуру аттестации.	**да**			
30.	Формы и методы контроля и оценки освоения ПК позволяют оценить сформированность ПК.	**да**			
31.	Основные показатели оценки результата позволяют однозначно диагностировать сформированность соответствующих общих компетенций (ОК).	**да**			
32.	Наименование форм и методов контроля и оценки освоения ОК точно и однозначно описывает процедуру аттестации.	**да**			
33.	Формы и методы контроля и оценки освоения ОК позволяют оценить сформированность ОК.	**да**			

Итоговое заключение (из трех альтернативных позиций следует выбрать одну)	**да**	**нет**
Программа профессионального модуля может быть рекомендована к утверждению	да	
Программу профессионального модуля следует рекомендовать к доработке		
Программу профессионального модуля следует рекомендовать к отклонению		

Замечания и рекомендации эксперта по доработке_____

Разработчик программы: _____ **Е.В. Волошин**
«22 апреля» 2020 г.

Эксперты: _____ **В.В. Неслюзов**
_____ **О.В. Бажин**
« 25 » апреля 2020 г.

МИНОБРНАУКИ РОССИИ

ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ «ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ЭКОНОМИКИ И СЕРВИСА» В Г. АРТЕМЕ

УТВЕРЖДАЮ
Заместитель директора
филиала
_____ О.И. ИВАНЮГА

КОНТРОЛЬНО-ОЦЕНОЧНЫЕ СРЕДСТВА ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

ПМ.01 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ

ПРОГРАММЫ ПОДГОТОВКИ СПЕЦИАЛИСТОВ СРЕДНЕГО ЗВЕНА
по специальности
Программирование в компьютерных системах

Уровень подготовки: базовый

Год набора на ООП
2020

Артем 2020

СОГЛАСОВАНО

Протокол заседания научно-методического совета
от 18 мая 2020г. №7

Председатель_____О.И. Иванюга

РАССМОТРЕНО И ОДОБРЕНО

на заседании кафедры ЭУИТ

Протокол № 14 от 06 мая 2020г.

И.о.зав.кафедрой _____ А.А. Власенко

Разработчик: _____ А.С. Бажина

Разработчик: Е.В.Волошин

Преподаватель кафедры ЭУИТ

Сожержание:

1. Общие положения

Результатом освоения профессионального модуля является готовность обучающегося к выполнению вида профессиональной деятельности **Разработка программных модулей программного обеспечения компьютерных систем.**

Формой итоговой аттестации по профессиональному модулю является **экзамен (квалификационный)**. Итогом этого экзамена является однозначное решение: «**Вид профессиональной деятельности освоен / не освоен».**

1. Формы контроля и оценивания элементов профессионального модуля

Таблица 1

Элемент модуля	**Форма контроля и оценивания**	
	Промежуточная аттестация	**Текущий контроль**
МДК.01.01 Системное программирование	Дифференцированный зачёт	- защита лабораторных и практических работ, внеаудиторных работ; - контрольные работы по темам МДК
МДК.01.02 Прикладное программирование	Дифференцированный зачёт	- защита лабораторных и практических работ, внеаудиторных работ; - контрольные работы по темам МДК
УП.01.01.	Дифференцированный зачёт	Защита по каждому из разделов МДК.
ПП.01.02	Дифференцированный зачёт	Защита по каждому из разделов МДК.
ПМ.01	Экзамен (квалификационный)	

2. Результаты освоения модуля, подлежащие проверке

2.1. Профессиональные и общие компетенции:

Таблица 2

Профессиональные и общие компетенции	Показатели оценки результата
ПК1.1. Выполнять разработку спецификаций отдельных компонент.	Создание отдельных компонент
	Выполнение спецификаций компонент
ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.	Выполнение создания кода программного продукта на уровне модуля в соответствии с требованиями к готовому программному продукту.
	Разработка пользовательского интерфейса.
ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств.	Использование специализированных программных средств отладки программных модулей.
	Разработка компонент программных модулей с использованием современных инструментальных средств и технологий.
ПК 1.4. Выполнять тестирование программных модулей.	Выполнение тестирования качества разработки программных модулей с помощью разработанных тестовых наборов и сценариев.
	Определение ошибок в программном коде с использованием тестовых наборов.
ПК 1.5. Осуществлять оптимизацию программного кода модуля	Выявление избыточности кода программного продукта и его оптимизация.
	Анализ оптимизации программного кода модуля.
ПК 1.6. Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций.	Использование инструментальных средств и графических языков спецификаций для создания компонент проектной и технической документации.
	Оформление проектной и технической документации в соответствии со стандартами.
ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.	Динамика успеваемости по МДК, положительный отзыв руководителя практики.
	Активное посещение учебных занятий и практики, консультаций.
ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество	Мотивированное обоснование выбора и применения методов и способов решения профессиональных задач.
	Точность, правильность и полнота выполнения профессиональных задач.
	Выполнение создания кода программного продукта на уровне модуля в соответствии с готовыми спецификациями
	Разработка пользовательского интерфейса

ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность	Демонстрация способности принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.
	Использование специализированных программных средств отладки программных модулей.
	Разработка компонент программных модулей с использованием современных инструментальных средств и технологий.
ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития	Обоснованность выбора информационных источников для решения профессиональных задач.
	Оперативность поиска и использования необходимой информации для качественного выполнения профессиональных задач, профессионального и личностного развития.
	Широта использования различных источников информации, включая электронные.
	Использование инструментальных средств и графических языков спецификаций для создания компонент проектной и технической документации.
	Оформление проектной и технической документации в соответствии со стандартами.
ОК 5. Владеть информационной культурой, анализировать и оценивать информацию с использованием информационно-коммуникационных технологий	Оперативность и широта осуществления операций с использованием общего и специализированного программного обеспечения.
	Создание отдельных компонент.
	Выполнение спецификаций компонент.
ОК 6. Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями	Результаты выполнения задания на учебной практике.
	Положительный отзыв руководителя практики.
	Выполнение создания кода программного продукта на уровне модуля в соответствии с готовыми спецификациями.
	Разработка пользовательского Интерфейса.
	Использование специализированных программных средств отладки программных модулей.
	Разработка компонент программных модулей с использованием современных инструментальных средств и технологий.
	Выполнение тестирования качества разработки программных модулей с помощью разработанных тестовых наборов и сценариев.
	Определение ошибок в программном коде с использованием тестовых наборов.
ОК 7. Брать на себя ответственность за работу членов команды, результат выполнения заданий	Ответственность за результат выполнения заданий на практике.
	Способность к самоанализу и коррекции результатов собственной работы.
	Использование специализированных программных средств отладки программных модулей.

	Разработка компонент программных модулей с использованием современных инструментальных средств и технологий.
	Выполнение тестирования качества разработки программных модулей с помощью разработанных тестовых наборов и сценариев.
	Определение ошибок в программном коде с использованием тестовых наборов.
	Выявление избыточности кода программного продукта и его оптимизация.
	Анализ оптимизации программного кода модуля.
ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации	Качество, своевременность и полнота выполнения заданий внеаудиторной самостоятельной работы.
	Обоснованность постановки целей и задач самообразования.
	Выполнение создания кода программного продукта на уровне модуля в соответствии с готовыми спецификациями.
	Разработка пользовательского Интерфейса.
	Использование специализированных программных средств отладки программных модулей.
	Разработка компонент программных модулей с использованием современных инструментальных средств и технологий.
	Выполнение тестирования качества разработки программных модулей с помощью разработанных тестовых наборов и сценариев.
	Определение ошибок в программном коде с использованием тестовых наборов.
	Выявление избыточности кода программного продукта и его оптимизация, анализ оптимизации программного кода модуля
ОК 9. Ориентироваться в условиях частой смены технологий в профессиональной деятельности	Проявление интереса к инновациям в области профессиональной деятельности
	Выполнение создания кода программного продукта на уровне модуля в соответствии с готовыми спецификациями
	Разработка пользовательского интерфейса
	Использование специализированных программных средств отладки программных модулей.
	Разработка компонент программных модулей с использованием современных инструментальных средств и технологий.
	Выполнение тестирования качества разработки программных модулей с помощью разработанных тестовых наборов и сценариев
	Определение ошибок в программном коде с использованием тестовых наборов.
	Выявление избыточности кода программного продукта и его оптимизация
	Анализ оптимизации программного кода модуля.

2.2. В результате изучения профессионального модуля обучающийся должен:

иметь практический опыт:

ПО 1. Разработка проекта программного продукта согласно постановке задачи пользователя.

ПО 2. Разработки кода программного продукта на основе готовой спецификации на уровне модуля.

ПО 3. Использования инструментальных средств на этапе отладки программного продукта.

ПО 4. Проведения тестирования программного модуля по определенному сценарию.

уметь:

У 1. Осуществлять разработку кода программного модуля на современных языках программирования.

У 2. Создавать программу по разработанному алгоритму как отдельный модуль.

У 3. Выполнять отладку и тестирование программы на уровне модуля.

У 4. Оформлять документацию на программные средства.

У 5. Использовать инструментальные средства для автоматизации оформления документации.

знать:

З 1. Основные этапы разработки программного обеспечения.

З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.

З 3. Основные принципы отладки и тестирования программных продуктов.

З 4. Методы и средства разработки технической документации

3. Оценка освоения теоретического курса профессионального модуля

3.1. Распределение типов контрольных заданий по элементам знаний и умений для оценки освоения МДК.01.01:

	У1	*У2*	*У3*	*У4*	*У5*	*З1*	*З2*	*З3*	*З4*
Тема 1.1 Интерфейс Windows приложений. Осуществление разработки кода программного модуля на языке С++	*ПЗ № 1,2,24-26,30* *ЛР № 1,5,6* *ВСР №1-2*	*ПЗ № 1,2,24-26,30* *ЛР № 1,5,6* *ВСР № 1,5,6*	*ПЗ № 1,2,24-26,30* *ЛР № 1,5,6* *ВСР № 1,5,6*			*В № 1-20*	*В № 21-75,91-119* *ПЗ № 32-35* *СР № 1-2*	*ЛР № 3,5,6* *ВСР № 3,5,6*	
Тема 1.2. Технология работы с файлами при создании модуля.	*ПЗ № 16-18* *ЛР № 7,16,17* *ВСР №7*	*ПЗ № 16-18* *ЛР № 7,16,17*	*ПЗ № 16-18* *ЛР № 7*				*В № 136-145, 199-207*		
Тема 1.3 Технология разработки модуля с использованием окон и элементов управления.	*ПЗ № 9-14,27,31* *ЛР № 2,3,8,10,14,15* *ВСР №3*	*ПЗ № 9-14,27,31* *ЛР № 2,3,8,11* *ВСР №8,11*	*ПЗ № 9-14,27,31* *ЛР № 2,3,8,10,14,15* *ВСР №10,14,15*				*В № 120-135*	*ЛР № 2,3*	
Тема 1.4 Растровая графика.	*ПЗ № 3-8,15,28,2*	*ПЗ № 3-8,15,28,2*	*ПЗ № 3-8,15,28,29*				*В № 76-90*	*ЛР № 4*	

Разработка модуля с элементами графики.	*9* *ЛР № 2,4,8,10, 11,13,19*	*9* *ЛР № 2,4,8,10,1 1,13,19*	*ЛР № 2,4,8,10,11 ВСР №11*						
Тема 1.5. Библиотека динамической компоновки DLL. Использование принципов структурного программирования	*ПЗ № 19-23 ЛР № 8,11,12, 17,18 ВСР №17,18*	*ПЗ № 19-23 ЛР № 8,11,12, 17,18*	*ПЗ № 19-23* *ЛР № 8,12 ВСР №12,13,16*				*В № 146-198 ПЗ № 32-35 СР № 1-2*	*ЛР № 2 ВСР №19 ,20*	
Тема 1.6 Методы и средства разработки технической документации программного продукта.				*ЛР № 20-23 ВСР 20-23*	*ЛР № 20-23*			*В № 208 ВСР 20-23*	*В № 209 ЛР № 20-23*

ПЗ – практическое задание
ЛР – лабораторная работа
ВСР – внеаудиторная самостоятельная работа
В – вопрос

3.1.1 Лабораторные работы

Лабораторная работа № 1
Тема: «Использование панели инструментов в WINAPI»
Цель: «Разработка кода программного модуля в WINAPI с использованием панели инструментов»
Задание (часть 1). Знакомство со средой программирования Visual C++, создание проекта, файла с исходным текстом в WINAPI, исполняемого модуля.
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Методические указания

Создание проекта Windows-приложения и Си-файла для его исходного кода:

- Создать проект Windows-приложения (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name -> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы.
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

Задание (часть 2). Создать простейшее приложение sample, которое выводит второе окно при нажатии правой кнопки мыши в клиентской области окна функции:
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)

Методические указания

Исходный текст приложения должен включать в себя:

```
// Нажата правая клавиша мыши в клиентской области окна.
        case WM_RBUTTONDOWN:
        {
                MessageBox(hWnd, "32-bit aplication", "Window",
                        MB_OK|MB_ICONINFORMATION);

        }; break;
```

Замечание. Текст приложения sample сохранить для дальнейшего его использования в качестве шаблона для других приложений

Лабораторная работа № 2

Тема: Стили классов окон, оконные стили в WINAPI

Цель: Разработка кода приложения с использованием оконных стилей.

Формулировка задания

Задание 1

На основе текста кода модуля в приложении 1, создать модуль который демонстрирует основные стили окон (окно верхнего уровня, всплывающее окно с родителем, всплывающее окно без родителя, дочернее окно).

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по	Осуществлена обработка событий	

разработанному алгоритму как отдельный модуль.		
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Методические указания

Для окон различных стилей (главное, всплывающее и дочернее) зарегистрировать отдельные классы окон ("MainWindows", "PopupWindows" и "ChildWindows"), предусмотрев для каждого класса собственный цвет фона и собственную функцию окна.

При создании окон название стиля окна вывести в заголовке окна, временные и дочерние окна расположить над поверхностью главного окна так, чтобы они не закрывали друг друга.

Для определения размера экрана используется функция GetSystemMetrics, а для определения рабочей области окна - функция GetClientRect. Определение расположения и размеров окна - функция GetWindowRect.

Замечание. Исходный текст приложения должен содержать подробные комментарии и объяснение поведения окон того или иного стиля.

Ход работы:

Использовать текст кода приложения № 1 ЛПЗ № 1.

1. Название класса главного окна разместить в переменной символьного типа ClassNameMW. Задать ей значение "MainWindow".

Заголовок главного окна разместить в переменной символьного типа AppTitleMW. Задать ей значение "Application MainWindow".

2. Название класса всплывающего окна разместить в переменной символьного типа ClassNamePP. Задать ей значение "PopupWindows".

Заголовок всплывающего окна разместить в переменной символьного типа AppTitlePP. Задать ей значение "Application PopupWindows".

3. Название класса дочернего окна разместить в переменной символьного типа ClassNameCH. Задать ей значение "ChildWindows".

Заголовок главного окна разместить в переменной символьного типа AppTitleCH. Задать ей значение "Application ChildWindows".

4. Дескрипторы главного, всплывающего, дочернего окна приложения разместить в переменных типа HWND с именами hWndMain, hWndPopup, hWndChild соответственно.
5. Кисть для главного окна выбрать серого цвета wc.hbrBackground=(HBRUSH)GetStockObject(GRAY_BRUSH);

6. Кисть для всплывающего окна выбрать красного цвета

wc.hbrBackground= CreateSolidBrush(RGB(255,0,0));

7. Кисть для дочернего окна выбрать бирюзового цвета

wc.hbrBackground= CreateSolidBrush(RGB(0,255,255));

8. Для главного, всплывающего и дочернего окон произвести:
   - заполнение структуры WNDCLASS для регистрации класса окна,
   - регистрацию класса окна,
   - создание окна,
   - отображение окна,
   - обновление содержимое клиентской области окна.

Пример для главного окна:

```
// Заполнение структуры WNDCLASS для регистрации класса окна.
        memset(&wc, 0, sizeof(wc));
        wc.lpszClassName=ClassNameMW;                          // Имя класса окон
        wc.lpfnWndProc=(WNDPROC)WndProc;                  // Адрес оконной функции
        wc.style=CS_HREDRAW|CS_VREDRAW;                          // Стиль класса
окон
        wc.hInstance=hInstance;                                  // Экземпляр приложения
        wc.hIcon=LoadIcon(NULL,IDI_APPLICATION);          // Пиктограмма для окон
        wc.hCursor=LoadCursor(NULL,IDC_ARROW);                  // Курсор мыши для
окон
        wc.hbrBackground=(HBRUSH)GetStockObject(GRAY_BRUSH);        // Кисть для окон
        wc.lpszMenuName=NULL;                                  // Ресурс меню окон
        wc.cbClsExtra=0;                                 // Дополнительная память
        wc.cbWndExtra=0;                                 // Дополнительная память

        // Регистрация класса окна.
        RegisterClass(&wc);
        // создание главного перекрывающегося окна
        hWndMain = CreateWindow(ClassNameMW,AppTitleMW,
                WS_OVERLAPPEDWINDOW,
                CW_USEDEFAULT,CW_USEDEFAULT,
CW_USEDEFAULT,CW_USEDEFAULT,
                NULL,NULL,hInstance,NULL);
           if(hWndMain == 0) return FALSE;
      ShowWindow(hWndMain, nCmdShow);
      UpdateWindow(hWndMain);
```

Аналогично создать всплывающее и дочернее окна.
Для всплывающего окна параметры функции CreateWindow задать

```
CW_USEDEFAULT,                    // X-координаты
CW_USEDEFAULT,                    // Y-координаты
CW_USEDEFAULT,                    // Ширина окна
CW_USEDEFAULT,                    // Высота окна
```

Равными 100,100,300,300 соответственно.

Для дочернего окна параметры функции CreateWindow задать

CW_USEDEFAULT, // X-координаты
CW_USEDEFAULT, // Y-координаты
CW_USEDEFAULT, // Ширина окна
CW_USEDEFAULT, // Высота окна

Равными 150,150,250,250 соответственно.
шаблона для других приложений

Лабораторная работа № 3
Тема: Стили классов окон, оконные стили **в WINAPI**. Окна с полосами прокрутки
Цель: Разработка кода модуля, содержащего создание окон с полосами прокрутки.
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Формулировка задания

Задание 1

На основе приложения 1, созданного на ЛПЗ № 2, создать приложение wstyle2, которое демонстрирует основные стили окон (окно верхнего уровня, всплывающее окно с родителем, всплывающее окно без родителя, дочернее окно). Главное окно создать с полосами вертикальной и горизонтальной прокрутки.

Методические указания

Для окон различных стилей (главное, всплывающее и дочернее) зарегистрировать отдельные классы окон ("MainWindows", "PopupWindows" и "ChildWindows"), предусмотрев для каждого класса собственный цвет фона и собственную функцию окна.

Добавить в окно приложения вертикальную или горизонтальную полосу прокрутки. Включить стиль WS_VSCROLL и WS_HSCROLL или оба сразу в описание стиля создаваемого функцией CreateWindow окна.

Замечание. Исходный текст приложения должен содержать подробные комментарии и объяснение поведения окон того или иного стиля.

Ход работы:

Каждая полоса прокрутки имеет соответствующий диапазон (range) – два целых числа, отражающих минимальное и максимальное значение, и положение (position) – местоположение бегунка внутри диапазона. По умолчанию устанавливается следующий диапазон прокрутки – минимум 0 и максимум 100, но диапазон легко изменить на любое другое значение с помощью функции SetScrollRange (для полос прокрутки окна hWndMain):

1. Создать статические переменные целого типа min_sb, max_sb,pos_sb
2. Задать им значения:

min_sb=1, max_sb=100, pos_sb=20;

3. После того, как зарегистрировано, создано, отображено и обновлено окно hWndMain обратиться к функции SetScrollRange, задав следующие параметры:

SetScrollRange(hWndMain,SB_VERT,min_sb,max_sb,TRUE);

4. Для установки нового положения бегунка использовать функцию SetScrollPos:

SetScrollPos(hWndMain,SB_VERT,pos_sb,TRUE);

5. В тело функции LRESULT CALLBACK WndProc(HWND hWndMain, UINT msg, WPARAM wParam, LPARAM lParam)

вставить следующий код для обработки сообщения WM_VSCROLL:

```
case WM_VSCROLL:
        {
                // запоминаем предыдущую позицию бегунка по вертикали
                int old_pos_sb=pos_sb;

                // lParam для оконных полос просмотра всегда равен NULL
                int nScrollCode=(int)LOWORD(wParam);      // произведенное действие
                short int nPos=(short int)HIWORD(wParam); // текущая позиция

                // изменеяем текущую позицию бегунка
                switch(nScrollCode)
                {
                        case SB_PAGEDOWN:       pos_sb+=10; break;
                        case SB_PAGEUP:         pos_sb-=10; break;
                        case SB_LINEDOWN:       pos_sb+=1; break;
                        case SB_LINEUP:         pos_sb-=1; break;
                        case SB_THUMBPOSITION:          pos_sb=nPos; break;
                        case SB_THUMBTRACK:   pos_sb=nPos; break;
                        default: return 0l;
                }
                if(pos_sb<min_sb)     pos_sb=min_sb;
                else if(pos_sb>max_sb)        pos_sb=max_sb;
```

```
        if(old_pos_sb!=pos_sb)
            SetScrollPos(hWndMain,SB_VERT,pos_sb,TRUE);
};
```

6. Отладить полученную программу.

Аналогично создать обработку сообщения WM_HSCROLL горизонтальной прокрутки окна hWndMain..

Лабораторная работа № 4
Тема: Обновление окна, вывод графики в окно в WINAPI.
Цель: Разработка кода модуля, содержащего обработку сообщения WM_PAINT для вывода текста и геометрических фигур в окно.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание (часть 1).
Знакомство со средой программирования Visual C++, создание проекта, файла с исходным текстом, исполняемого модуля.

Методические указания

Создание проекта Windows-приложения и Си-файла для его исходного кода:

- Создать проект Windows-приложения (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name "*Lab_3" *-> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы.
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

исходный текст программы взять из приложения № 1:Задание. (Часть 2)

На основе приложения создать приложение picture, обеспечивающее при получении сообщения WM_PAINT вывод некоторого изображения в окно.

Методические указания

- вызвать функцию **GetDC для получения дескриптора контекста устройства**, а для его освобождения – функцию **ReleaseDC**.
- Как правило, **вызовы функций GetDC и ReleaseDC используются в ответ на сообщения от клавиатуры или на сообщения от манипулятора мышь.** Они позволяют обновлять рабочую область непосредственно в ответ на пользовательский ввод информации.
- Псевдокод для отрисовки изображения и структурный тип, описывающий его переменные характеристики, приводятся **для каждого варианта задания**. Следующие характеристики являются обязательными для всех вариантов: ***местоположение, размер*** и ***цвет*** изображения.
- Список сообщений, обязательных для обработки функцией главного окна: WM_PAINT, WM_DESTROY. Остальные сообщения передать на обработку стандартной оконной функции.

Замечание. Текст выполняемого варианта приложения picture сохранить для дальнейшего его использования в качестве исходного для выполнения последующих заданий.

Ход работы:

1. Для вывода сообщения в центре главного окна в тело процедуры

LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)

вставить следующий код для обработки сообщения **WM_PAINT с помощью получения дескриптора контекста устройства** вызовом функции **GetDC**, а для его освобождения – функцию **ReleaseDC**:

```
// Необходимо обновить содержимое клиентской области окна.
        case WM_PAINT:
        {
        // Получить контекст окна
                HDC hDC=GetDC(hWnd);
                // получение размеров рабочей области окна
    RECT r; GetClientRect(hWnd,&r);
// получение информации о текущем шрифте
    TEXTMETRIC tm; GetTextMetrics(hDC,&tm);

    // координаты X и Y
    int x=(r.right-tm.tmAveCharWidth*strlen(str))/2;
    int y=(r.bottom-tm.tmHeight)/2;
```

```
// вывод строкиY
TextOut(hDC,x,y,str,strlen(str));

        // Освободить контекст окна
        ReleaseDC(hWnd,hDC);
    // Освободить контекст окна
  }; return 0;
```

2. Запустить программу и убедиться, что в центре главного окна выводится текст.
3. Для вывода в главном окне изображения в виде квадрата зелёного цвета после вывода строки вставить код:

```
// создать перо - сплошная зеленая линия
HPEN hPen=CreatePen(PS_SOLID,1,RGB(0,255,255));
// выбрать перо в контекст отображения
HPEN hOldPen=(HPEN)SelectObject(hDC,hPen);

// осуществить вывод изображения
Rectangle(hDC,300,300,100,50);
// восстановить в контексте "старое" перо

SelectObject(hDC,hOldPen);
// удалить созданное ранее перо
DeleteObject(hPen);
```

4. Запустить программу и убедиться, что в главном окне выводится квадрат зелёного цвета.
5. Изменить цвет и размеры квадрата и запустить программу.

Лабораторная работа № 5

Тема: Обработка клавиатурных сообщений в WINAPI
Цель: Разработка кода модуля, содержащего обработку клавиатурных сообщений
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки	Соответствие текста кода	

программного продукта.	программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание (часть 1).

Знакомство со средой программирования Visual C++, создание проекта, файла с исходным текстом, исполняемого модуля.

Методические указания

Создание проекта Windows-приложения и Си-файла для его исходного кода:

- Создать проект Windows-приложения (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name "*Lab_4-1" *-> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы.
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

.

исходный текст программы взять из приложения № 1:

Задание. (Часть 2)

На основе приложения создать приложение, обеспечивающее при получении клавиатурных сообщений WM_KEYDOWN выводить окно о нажатых клавишах "Space+Shift" и "<Ctrl> + <F1>" .

Методические указания

Использовать функцию **GetAsyncKeyState**. **Для получения информаци о текущем положении клавиши.**

Замечание. Текст выполняемого варианта приложения picture сохранить для дальнейшего его использования в качестве исходного для выполнения последующих заданий.

Ход работы:

6. Для обработки клавиатурных сообщений в тело процедуры

LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)

вставить следующий код:

```
case WM_KEYDOWN:
{       int nVirtKey = (int) wParam; // виртуальный код клавиши
        switch(nVirtKey)
        {
                // нажатие клавиши Space (пробел)
```

```
                case VK_SPACE:
                        // состояние на момент события
                        {if(GetKeyState(VK_SHIFT)<0)
                                MessageBox(hWnd,"Одновременно
Space+Shift","Space+Shift",MB_OK);}; break;
                // нажатие клавиши F1 для определения тек. состояния др. клавиши
                case VK_F1:
                        // текущее состояние
                        {if(GetAsyncKeyState(VK_CONTROL))
                                MessageBox(hWnd,"Текущее состояние <Ctrl> + <F1> -
нажата", "<F1>",MB_OK);};break;
                }
        }; return 0;
```

7. Запустить программу и убедиться, что при нажатии клавиш "Одновременно Space+Shift" появится окно с сообщением ,"Одновременно Space+Shift".
8. Запустить программу и убедиться, что при нажатии клавиш "<Ctrl> + <F1> " появится окно с сообщением ,"Текущее состояние <Ctrl> + <F1> - нажата".

Лабораторная работа № 6

Тема: Обработка сообщений мыши, связанных с рабочей областью окна.
Цель: Разработка кода модуля, содержащего обработку сообщений мыши, связанных с рабочей областью окна.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание (часть 1).

Знакомство со средой программирования Visual C++, создание проекта, файла с исходным текстом, исполняемого модуля.

Методические указания

Создание проекта Windows-приложения и Си-файла для его исходного кода:

- Создать проект Windows-приложения (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name "*Lab_4-1" *-> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы.
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

. исходный текст программы взять из приложения № 1:

Задание. (Часть 2)

На основе приложения создать приложение,

Методические указания

Определить присутствие мыши можно с помощью функции GetSystemMetrics, передав ей в качестве параметра значение **SM_MOUSEPRESENT**. Если мышь есть, эта функция возвращает ненулевое значение.

Для определения количества кнопок можно использовать вызов **GetSystemMetrics** с параметром **SM_CMOUSEBUTTONS**.

Замечание. Текст выполняемого варианта приложения picture сохранить для дальнейшего его использования в качестве исходного для выполнения последующих заданий.

Ход работы:

1. Для проверки, была ли нажата левая кнопка мыши при ее движении по рабочей области окна, и изображения на этом месте пиксела, в тело процедуры

LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)

вставить следующий код:

```
// Нажата левая клавиша мыши в клиентской области окна.

        case WM_MOUSEMOVE:
{
        // состояние кнопок мыши
        UINT fwKeys = wParam;
        // горизонтальная позиция курсора
        int xPos = LOWORD(lParam);
        // вертикальная позиция курсора
        int yPos = HIWORD(lParam);

        if(fwKeys & MK_LBUTTON)
        {
                HDC hDC=GetDC(hWnd);
```

```
                SetPixel(hDC,xPos,yPos,0l);
                ReleaseDC(hWnd,hDC);
        }
}; return 0;
```

- Запустить программу и убедиться, что при нажатии левой кнопки мыши в клиентской части главного окна изображается чёрная точка.

Лабораторная работа № 7

Тема: Основы использования таймера **в WINAPI**
Цель: Разработка кода модуля, содержащего обработку сообщений таймера.
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание (часть 1).
Знакомство со средой программирования Visual C++, создание проекта, файла с исходным текстом, исполняемого модуля.
Методические указания
Создание проекта Windows-приложения и Си-файла для его исходного кода:

- Создать проект Windows-приложения (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип*

*создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name "*Lab_4-1" *-> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы.
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

.

исходный текст программы взять из приложения № 1:

Задание. (Часть 2)

На основе приложения 1 создать приложение, обрабатывающее сообщения от таймера.

Методические указания

Таймер в Windows является устройством ввода информации, которое периодически извещает приложение о том, что истек заданный интервал времени.

Приложение сообщает системе Windows интервал времени, а затем Windows периодически посылает приложению сообщения **WM_TIMER**, сигнализируя об истечении интервала

времени. **Присоединить таймер к программе можно при помощи вызова функции SetTimer.** Функция **SetTimer** содержит целый параметр, задающий интервал в миллисекундах – это значение определяет темп, с которым Windows посылает программе сообщения **WM_TIMER**.

- **Для остановки потока сообщений от таймера приложение должно вызвать функцию KillTimer**. Вызов **KillTimer** очищает очередь сообщений от всех необработанных сообщений **WM_TIMER**.

- Поскольку приложения Windows **получают сообщения WM_TIMER из обычной очереди сообщений**, приложение **не должно беспокоиться** о том, что его работа будет прервана внезапным сообщением **WM_TIMER**.
- **В этом смысле таймер похож на клавиатуру и мышь:** драйвер обрабатывает асинхронные аппаратные прерывания, а Windows преобразует эти прерывания в регулярные, структурированные, последовательные сообщения.

Итак, сообщения таймера ставятся в обычную очередь сообщений и обрабатываются как все остальные сообщения. Поэтому, если приложение задает функции **SetTimer** интервал 1000 миллисекунд, то **ему не гарантируется** получение сообщения каждую секунду (интервал будет колебаться). Если приложение занято больше чем секунду, то оно вообще не получит ни одного сообщения **WM_TIMER** в течение этого времени. Фактически,

- **Windows обрабатывает сообщение WM_TIMER во многом так же, как сообщения WM_PAINT**. Оба эти сообщения имеют **низкий приоритет**, и программа получит их, только если **в очереди нет других сообщений**.
- Сообщения **WM_TIMER** похожи на сообщения **WM_PAINT** и в другом смысле: **Windows не хранит в очереди сообщений несколько сообщений WM_TIMER**. Вместо этого Windows объединяет несколько сообщений **WM_TIMER** из очереди в одно сообщение. В результате **приложение не может определить число "потерянных" сообщений WM_TIMER**.

Таймер можно использовать одним из трех способов, в зависимости от параметров функции **SetTimer.**

Вызов функции **SetTimer** в этом случае обычно имеет примерно такой вид:

```
SetTimer(hWnd,1,1000,NULL);
```

- Первый параметр – дескриптор того окна, чья **оконная процедура будет получать сообщения WM_TIMER**.
- Вторым параметром является **идентификатор таймера**, значение которого должно быть отлично от нуля. В этом примере он произвольно установлен в 1.
- Третий параметр – это 32-разрядное беззнаковое целое, которое задает **интервал в миллисекундах** (значение 1000 задает генерацию сообщений **WM_TIMER** один раз в секунду).

Замечание. Текст выполняемого варианта приложения picture сохранить для дальнейшего его использования в качестве исходного для выполнения последующих заданий.

Ход работы:

1. Задать команды препроцессора:

```
#define TIMER_SEC 1
#define TIMER_MIN 2
```

2. В тело функции WndProc

LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)

вставить обработку событий:

3. При нажатии левой клавишы мыши в клиентской области окна вызвать функцию SetTimer со следующими параметрами:

```
SetTimer(hWnd,TIMER_SEC,1000,NULL);
```

4. При нажатии правой клавишы мыши в клиентской области окна вызвать функцию SetTimer со следующими параметрами:

```
SetTimer(hWnd,TIMER_MIN,6000,NULL);
```

5. При нажатии клавишы CONTROL в клиентской области окна вызвать функцию KillTimer для остановки таймера:

```
KillTimer(hWnd,TIMER_SEC);
```

6. При нажатии клавишы SHIFT в клиентской области окна окна вызвать функцию KillTimer для остановки таймера:

```
KillTimer(hWnd,TIMER_MIN);
```

7. вставить следующий код обработка сообщений таймера

```
case WM_TIMER:
{   UINT wTimerID = wParam;
    switch(wTimerID)
    {
        // действия происходят один раз в секунду
        case TIMER_SEC:
            {MessageBox(hWnd,"действия происходят один раз в секунду","Таймер 1 сек",MB_OK);

            break;}
        // действия происходят один раз в минуту
```

```
                case TIMER_MIN:
                        {MessageBox(hWnd,"действия происходят один раз в
минуту","Таймер 1 мин",MB_OK);
                        break;}
            }; return 0;
        }
```

- Запустить программу и убедиться, что при нажатии левой и правой кнопок мыши в клиентской части главного окна запускается окно с соответствующим сообщением согласно заданной обработке сообщений таймера.

Лабораторная работа № 8

Тема: Программирование для Windows – интерфейс Win API.

Приложение **WINAPI** с использованием элементов управления.

Цель: Разработка кода приложения с использованием элементов управления.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.

За не полностью выполненную работу выставляется – 1 балл.

За невыполненную работу выставляется – 0 баллов.

Формулировка задания:

Задание 1

На основе текста кода модуля в приложении 1, создать модуль Lab_5-1, который создаёт кнопку и редактор с помощью элементов управления button, edit.

Методические указания

 Любое стандартное приложение Windows использует различные элементы управления, такие, как кнопки, полосы просмотра, редакторы текстов и т.д, реализованные в виде дочерних окон.

Каждое дочернее окно создается с помощью вызова функции CreateWindow. Оконная процедура родительского окна посылает сообщения дочерним окнам управления, а дочерние окна управления посылают сообщения обратно оконной процедуре.

Если создается одно из предопределенных дочерних окон управления, то для этого дочернего окна класс окна регистрировать не надо. Такой класс уже существует в Windows и имеет одно из следующих имен: “button”, ”edit”, “static”, “listbox”, “combobox” и “scrollbar”. Приложение просто использует одно из этих имен в качестве параметра в функции CreateWindow.

Пример:

```
static UINT ID_button=1;
static HWND hWndButton;
        . . .
hWndButton=CreateWindow("button","Exit",
                WS_CHILD|WS_VISIBLE|WS_CLIPSIBLINGS|BS_PUSHBUTTON,
                100,400,50,25,hWnd,(HMENU)ID_button,hInst,NULL);
        . . .
```

В оконной процедуре родительского окна, в котором происходит обработка нажатия на дочернее окно-кнопку с идентификатором ID_button вставить код:

```
        case WM_COMMAND:
        {       UINT idCtl=LOWORD(wParam); // идентификатор дочернего окна
                UINT code=HIWORD(wParam);  // код уведомления
                HWND hChild=(HWND)lParam;  // дескриптор дочернего окна
                if(idCtl==ID_button&&code==BN_CLICKED)
                {
                        // кнопка была нажата
                        //CloseWindow(hWnd); // закрыть окно-родителя
                                PostQuitMessage(0); //закрыть приложение
                }
        }; return 0;
```

Для того, чтобы создать свой собственный редактор, достаточно создать на базе класса “edit” окно управления, вызвав функцию CreateWindow, например:

```
        static UINT ID_edit=3;
        static HWND hWndEdit;
        . . .
        hWndEdit=CreateWindow("edit",NULL,
                WS_CHILD|WS_VISIBLE|WS_CLIPSIBLINGS| WS_BORDER|ES_LEFT,
                100,450,50,25,hWnd,(HMENU)ID_edit,hInst,NULL);
```

Для того, чтобы после создания редактора, поместить в редактируемое поле текст, можно воспользоваться функцией SetWindowText.

```
char str_a[]="     ";
SetWindowText(hWndEdit,str_a); // в текстовом редакторе вывести содержимое переменной str_a
SetWindowText(hWndEdit,"      ");// в текстовом редакторе вывести 6 пробелов
```

Для получения текста из окна редактирования используются функции GetWindowTextLength и GetWindowText.

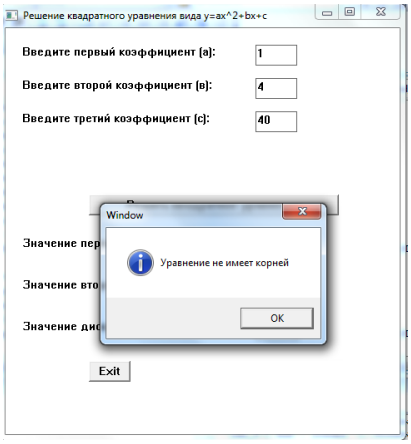
Пример:

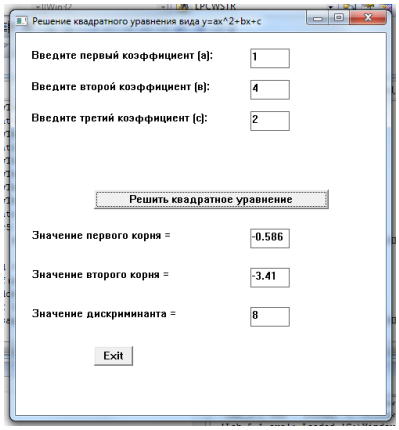
```
char str_a[]="     ";
                    ……….

                    GetWindowText(hWndEdit,str_a,6); //получить в переменную str_a текст из редактора с дескриптором  hWndEdit
                    float a = atof(str_a); // функция преобразования текстовой строки в целое число
                    MessageBox(hWnd, str_a, "Window",
                            MB_OK|MB_ICONINFORMATION);
```

- *Замечание.* Исходный текст приложения должен содержать подробные комментарии и объяснение поведения окон того или иного стиля.

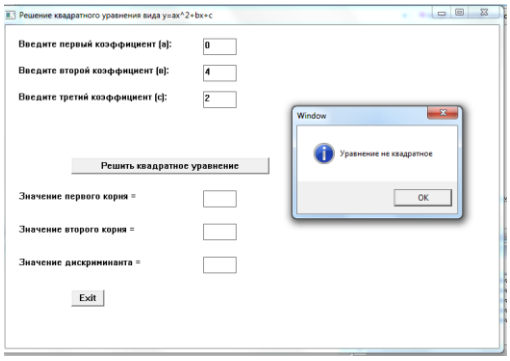
Задание 2

На основе текста кода модуля в приложении 1, создать модуль, который выводит форму для решения квадратного уравнения y = ax^2+bx+c.
Коэффициенты a, в, с вводятся с помощью элементов управления edit.
Для начала решения квадратного уравнения используется элемент управления button.
Для завершения работы приложения используется элемент управления button.
Входные данные: коэффициенты a, в, с
Выходные данные: корни уравнения x1, x2.
D=b*b-4*a*c (D - дискриминант)
Если дискриминант уравнения отрицательный, выдать сообщение: «Нет корней»,
Иначе: x1=(-b+sqrt(D))/(2*a);   x2=(-b-sqrt(D))/(2*a) .
Ход работы:
Создать приложение, выводящее форму:





Если дискриминант уравнения отрицательный, выдать сообщение:
Если коэффициент a=0, то вывести сообщение "Уравнение не квадратное"



Лабораторная работа № 10

Тема: Программирование для Windows – интерфейс Win32 API (Application Programming Interface)..

Приложение с использованием элементов управления

Цель: Разработка кода приложения с использованием элементов управления.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Формулировка задания:

Задание 1

На основе текста кода модуля в приложении 1, создать модуль, который создаёт кнопки и редакторы с помощью элементов управления button, edit.

Методические указания

Любое стандартное приложение Windows использует различные элементы управления, такие, как кнопки, полосы просмотра, редакторы текстов и т.д, реализованные в виде дочерних окон.

Каждое дочернее окно создается с помощью вызова функции CreateWindow. Оконная процедура родительского окна посылает сообщения дочерним окнам управления, а дочерние окна управления посылают сообщения обратно оконной процедуре.

Если создается одно из предопределенных дочерних окон управления, то для этого дочернего окна класс окна регистрировать не надо. Такой класс уже существует в Windows и имеет одно из следующих имен: “button”, ”edit”, “static”, “listbox”, “combobox” и “scrollbar”. Приложение просто использует одно из этих имен в качестве параметра в функции CreateWindow.

Пример:

```
static UINT ID_button=1;
static HWND hWndButton;
        . . .
hWndButton=CreateWindow("button","Exit",
                        WS_CHILD|WS_VISIBLE|WS_CLIPSIBLINGS|BS_PUSHBUTTON,
                        100,400,50,25,hWnd,(HMENU)ID_button,hInst,NULL);
        . . .
```

В оконной процедуре родительского окна, в котором происходит обработка нажатия на дочернее окно-кнопку с идентификатором ID_button вставить код:

```
        case WM_COMMAND:
        {       UINT idCtl=LOWORD(wParam); // идентификатор дочернего окна
                        UINT code=HIWORD(wParam);  // код уведомления
                        HWND hChild=(HWND)lParam;  // дескриптор дочернего окна
                        if(idCtl==ID_button&&code==BN_CLICKED)
                        {
                                // кнопка была нажата
                                //CloseWindow(hWnd); // закрыть окно-родителя
                                   PostQuitMessage(0); //закрыть приложение
                        }
        }; return 0;
```

++

Для того, чтобы создать свой собственный редактор, достаточно создать на базе класса “edit” окно управления, вызвав функцию CreateWindow, например:

```
        static UINT ID_edit=3;
        static HWND hWndEdit;
        . . .
        hWndEdit=CreateWindow("edit",NULL,
                        WS_CHILD|WS_VISIBLE|WS_CLIPSIBLINGS| WS_BORDER|ES_LEFT,
                        100,450,50,25,hWnd,(HMENU)ID_edit,hInst,NULL);
```

Для того, чтобы после создания редактора, поместить в редактируемое поле текст, можно воспользоваться функцией SetWindowText.

```
char str_a[]=" ";
SetWindowText(hWndEdit,str_a); // в текстовом редакторе вывести содержимое переменной str_a
SetWindowText(hWndEdit,"      ");// в текстовом редакторе вывести 6 пробелов
```

++
Для получения текста из окна редактирования используются функции GetWindowTextLength и GetWindowText.
Пример:

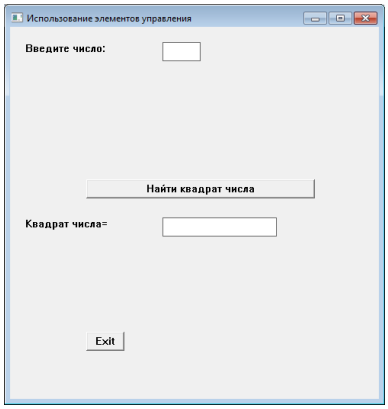
```
char str_a[]=" ";
                    …….

                    GetWindowText(hWndEdit,str_a,6); //получить в переменную str_a текст из редактора с дескриптором  hWndEdit
                    float a = atof(str_a); // функция преобразования текстовой строки в целое число
                    MessageBox(hWnd, str_a, "Window",
                            MB_OK|MB_ICONINFORMATION);
```

Задание 2

На основе текста кода модуля в приложении 1, создать модуль, который выводит форму для нахождения квадрата числа.
Число вводится с помощью элемента управления edit.
Для начала вычисления квадрата числа используется элемент управления button.
Для завершения работы приложения используется элемент управления button.
Входные данные: число a
Выходные данные: квадрат числа a

Ход работы:

Для создания приложения, выводящего форму:



1. Описать статические переменные:

```
static UINT ID_button=1; // кнопка выхода из приложения
static HWND hWndButton; //

static UINT ID_edit2=3; // редактор для ввода числа
static HWND hWndEdit2; //

static UINT ID_button5=5; // кнопка команды для вычисления квадрата числа
static HWND hWndButton5; //

static HWND hWndEdit_x1; // редактор для вывода квадрата числа
static UINT ID_edit_x1=6; //
```

2. После обновления содержимого клиентской области окна функцией UpdateWindow(hWnd);

Создать элементы управления:

```
hWndEdit2=CreateWindow("edit",NULL,
                WS_CHILD|WS_VISIBLE|WS_CLIPSIBLINGS| WS_BORDER|ES_LEFT,
                300,20,50,25,hWnd,(HMENU)ID_edit2,hInst,NULL);

hWndEdit_x1=CreateWindow("edit",NULL,
                WS_CHILD|WS_VISIBLE|WS_CLIPSIBLINGS| WS_BORDER|ES_LEFT,
                300,250,150,25,hWnd,(HMENU)ID_edit_x1,hInst,NULL);

hWndButton=CreateWindow("button","Найти квадрат числа",
                WS_CHILD|WS_VISIBLE|WS_CLIPSIBLINGS|BS_PUSHBUTTON,
                100,200,300,25,hWnd,(HMENU)ID_button5,hInst,NULL);
hWndButton=CreateWindow("button","Exit",
                WS_CHILD|WS_VISIBLE|WS_CLIPSIBLINGS|BS_PUSHBUTTON,
                100,400,50,25,hWnd,(HMENU)ID_button,hInst,NULL);
```

3. В тело функции

LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)

Вставить описание переменных:

```
        char  *str="Введите число: ";
char *str_x1="Квадрат числа= ";
```

и следующий код для обработки сообщения WM_COMMAND::

```
char str_a[]="     ";

case WM_COMMAND:
        {       UINT idCtl=LOWORD(wParam); // идентификатор дочернего окна
                UINT code=HIWORD(wParam);  // код уведомления
                HWND hChild=(HWND)lParam;  // дескриптор дочернего окна
                if(idCtl==ID_button&&code==BN_CLICKED)
                {
                        // кнопка была нажата
                        //CloseWindow(hWnd); // закрыть окно-родителя
                        PostQuitMessage(0);  // закрыть приложение
                }
// кнопка была нажата
                if(idCtl==ID_button5&&code==BN_CLICKED)
                {
                        // кнопка была нажата
                        GetWindowText(hWndEdit2,str_a,4); // получить в переменную str_a текст из редактора с дескриптором  hWndEdit2
                        float a = atof(str_a); // функция преобразования текстовой строки в целое число

                float x1 = a * a; // функция преобразования текстовой строки в целое число
```

```
                char* str5 = new char[30]; //
        sprintf(str5, "%.3g", x1 ); // перевести число x1 в строку str5
                SetWindowText(hWndEdit_x1,str5); //вывести строку str5 в редакторе
hWndEdit_x1

                }
        }; return 0;
```

Запустить приложение и отладить.

Задание 3:

Изменить приложение согласно постановке задачи:

Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b.

Вычислить длину окружности и площадь круга одного и того же заданного радиуса *R*.

Найти произведение цифр заданного четырехзначного числа.

Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.

Даны два действительных числа X и Y. Вычислить их сумму, разность, произведение и частное.

Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.

Дана сторона равностороннего треугольника. Найти площадь этого треугольника, его высоты, радиусы вписанной и описанной окружностей.

Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.

Найти площадь кольца, внутренний радиус которого равен r, а внешний – заданному числу R(R>r).

Треугольник задан величинами своих углов, и радиусом описанной окружности. Найти стороны треугольника.

Найти площадь равнобедренной трапеции с основаниями a и b и углом α при большем основании a.

Лабораторная работа № 11

Тема: Создание меню

Цель: Разработка кода модуля, содержащего обработку сообщения **WM_COMMAND** для вывода пунктов меню.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**

У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание 1.

Знакомство со средой программирования Visual C++, создание проекта, файла с исходным текстом, исполняемого модуля.

Методические указания

Создание проекта Windows-приложения и Си-файла для его исходного кода:

- Создать проект Windows-приложения (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name "*Lab_3" -> *выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы.
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

исходный текст программы взять из приложения № 1:

Методические указания

Динамическое создание меню

создание меню верхнего уровня, элементами которого являются два всплывающих меню (каждое содержит по два пункта) и один обычный элемент, и последующего его присоединения к уже созданному окну hWnd.

Для **создания всплывающего меню (подменю)** существует функция **CreatePopupMenu**:

```
static HMENU hMenuPopup;
```

. . .
hMenuPopup=CreatePopupMenu();

К созданному пустому всплывающему меню **можно добавить элементы - пункты или всплывающие меню (вложенные)**:

AppendMenu(**hMenuPopup**,MF_STRING,ID_ITEM1,”Текст пункта 1 вспл. Меню”);

AppendMenu(**hMenuPopup**,MF_STRING,ID_ITEM2,”Текст пункта 2 вспл. Меню”);

В приведенном выше примере было создано всплывающее меню с двумя пунктами. Идентификаторы ID_ITEM1 и ID_ITEM2 этих пунктов предварительно следует определить, например, при помощи директивы #define.

В дальнейшем можно **добавить в меню верхнего уровня созданные функцией CreatePopupMenu всплывающие меню или отдельные пункты**, вызвав функцию **AppendMenu**:

AppendMenu(**hMenu**,MF_POPUP,(UINT)hMenuPopup,”Текст заголовка вспл. Меню”);

AppendMenu(**hMenu**,MF_STRING,ID_ITEM3,”Текст пункта главного меню”);

Идентификатор добавляемого пункта ID_ITEM3 также должен быть заранее определен, например, при помощи директивы #define.

Затем следует **подключить меню верхнего уровня к созданному ранее окну** hWnd функцией **SetMenu** и отобразить новое меню окна функцией **DrawMenuBar**:

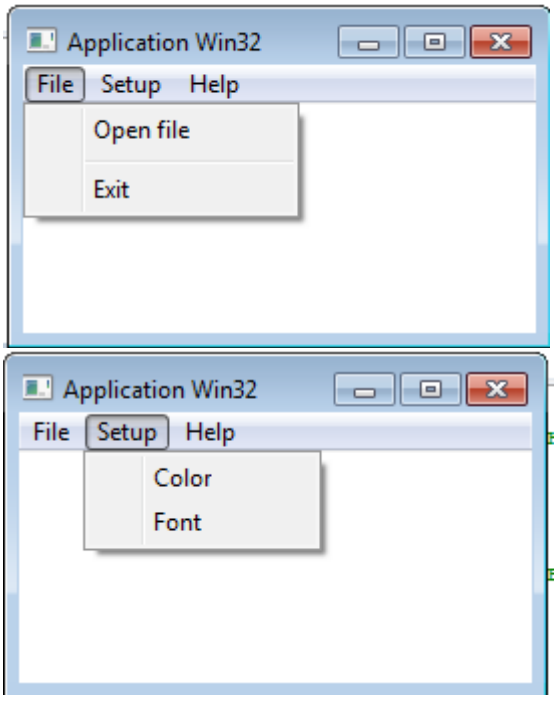
SetMenu(hWnd,**hMenu**);

DrawMenuBar(hWnd);

Задание:

- Требуется создать динамически подключаемое меню с помощью вызовов функций CreateMenu и AppendMenu. После завершения определения меню передать дескриптор меню функции CreateWindow и использовать функцию SetMenu для установки меню окна.
- Меню верхнего уровня: File, Setup, Help
- Пункты меню File: Open file, Exit
- Пункты меню Setup: Color, Font

.



Ход работы

1. Определить при помощи директивы #define идентификаторы ID_OPEN, ID_EXIT, ID_COLOR, ID_FONT, ID_HELP

```
#define ID_OPEN     40
#define ID_EXIT     41
#define ID_COLOR    42
#define ID_FONT     43
#define ID_HELP     44
```

2. Описать переменные hMainMenu, hMenuFile, hMenuSetup типа static HMENU.
3.

```
static HMENU hMainMenu, hMenuFile, hMenuSetup;
```

4. с помощью вызовов функций **CreateMenu** и **AppendMenu** создать **все меню в процессе работы приложения**. После завершения определения меню использовать функцию **SetMenu** для установки меню окна.

```
hMainMenu=CreateMenu();

        hMenuFile=CreatePopupMenu();
        AppendMenu(hMenuFile,MF_STRING,ID_OPEN,"Open file");
        AppendMenu(hMenuFile,MF_SEPARATOR,0,NULL);
    AppendMenu(hMenuFile,MF_STRING,ID_EXIT,"Exit");

        hMenuSetup=CreatePopupMenu();
        AppendMenu(hMenuSetup,MF_STRING,ID_COLOR,"Color");
        AppendMenu(hMenuSetup,MF_STRING,ID_FONT,"Font");

        AppendMenu(hMainMenu,MF_POPUP,(UINT)hMenuFile,"File");
        AppendMenu(hMainMenu,MF_POPUP,(UINT)hMenuSetup,"Setup");
        AppendMenu(hMainMenu,MF_STRING,ID_HELP,"Help");
```

5. Затем **подключить меню верхнего уровня к созданному ранее окну** hWnd функцией **SetMenu** и отобразить новое меню окна функцией **DrawMenuBar**:

```
        SetMenu(hWnd,hMainMenu);
        DrawMenuBar(hWnd);
```

6. Отличить сообщения **WM_COMMAND**, пришедшие от меню, от сообщений **WM_COMMAND**, присылаемые **элементами управления**, в случае, если они имеют одинаковые идентификаторы, можно при помощи значения параметра lParam:
   для **пунктов меню** он всегда равен 0,
   для **элементов управления** он равен дескриптору окна элемента управления.

В тело оконной функции включить следующий код:

```
case WM_COMMAND:
        {       UINT idCtl=LOWORD(wParam); // идентификатор дочернего окна
                UINT code=HIWORD(wParam);  // код уведомления
                HWND hChild=(HWND)lParam;  // дескриптор дочернего окна

                if(lParam!=0)
                {
```

```
                // сообщение от элемента управления - другой способ обработки
                        }
        else
        {
                UINT idItem= LOWORD(wParam); // идентификатор пункта меню
                switch(idItem)
                {
                        case ID_OPEN:
                                MessageBox(hWnd, " Выбран пункт меню open", "Window",
                        MB_OK|MB_ICONINFORMATION);// пункт меню выбран
                                break;
                        case ID_EXIT:
                                PostQuitMessage(0);  // закрыть приложение
                                break;
                        case ID_COLOR:
                                MessageBox(hWnd, " Выбран пункт меню color",
"Window",
                        MB_OK|MB_ICONINFORMATION); // пункт меню выбран
                                break;
                                case ID_FONT:
                                MessageBox(hWnd, " Выбран пункт меню font", "Window",
                        MB_OK|MB_ICONINFORMATION); // пункт меню выбран
                                break;
          case ID_HELP:
        MessageBox(hWnd, "Выбран пункт меню Help", "Window",
                        MB_OK|MB_ICONINFORMATION); // пункт меню выбран
                                break;
                }
          }

    }; return 0;
```

7. Запустить программу и убедиться, что в главном окне появилось меню и для каждого пункта меню выводится соответствующее сообщение.
8. Разработать приложение, создающее динамически подключаемое меню с названием «Формулы», состоящее из пунктов:

Площадь	**Объём**	**Теоремы**	**Тригонометрия**
Прямоугольника	**Цилиндра**	**Пифагора**	**Sin(x)**
Треугольника	**Конуса**		**Cos(x)**
Трапеции	**Шара**		**Tg(x)**
Круга	**Пирамиды**		**Ctg(x)**
	Параллелограмма		**Триг. тождество**

Лабораторная работа № 12

Тема: Меню. Диалоговые окна общего пользования

Цель: Разработка кода модуля, содержащего вывод диалоговых окон:

- диалоговые окна открытия (“Open”) и сохранения (“Save As”) файлов,

- выбора цвета (“Color”) и шрифта (“Font”)

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание 1.

Знакомство со средой программирования Visual C++, создание проекта, файла с исходным текстом, исполняемого модуля.

Методические указания

Создание проекта Windows-приложения и Си-файла для его исходного кода:

- **Создать проект Windows-приложения (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name* “Lab_3” *-> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).**

Создание исполняемого модуля Си++-программы для Windows:

- **Ввести в поле редактирования файла исходный текст программы.**
- **Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;**
- **Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).**

.

исходный текст программы взять из приложения № 1:

Задание 2

- **На основе приложения 1 создать приложение picture, обеспечивающее при получении сообщения WM_COMMAND вывод диалоговых окон открытия (“Open”) и сохранения (“Save As”) файлов, выбора цвета (“Color”) и шрифта (“Font”).**

.

Методические указания

Диалоговое окно “Open” открывается при вызове функции GetOpenFileName. Единственным параметром функции является указатель на структуру OPENFILENAME. Члены этой структуры обеспечивают значения для инициализации диалогового окна (и, необязательно, адрес обработчика и имя шаблона пользовательского диалогового окна приложения, которые используются для настройки диалогового окна).

Диалоговое окно “Save As” отображается в ответ на вызов функции GetSaveFileName. Эта функция также имеет единственный параметр - указатель на структуру OPENFILENAME.

- Диалоговое окно “Color” используется для выбора цвета пользователем. Это диалоговое окно используется для выбора цвета из системной палитры или определения пользовательского цвета.

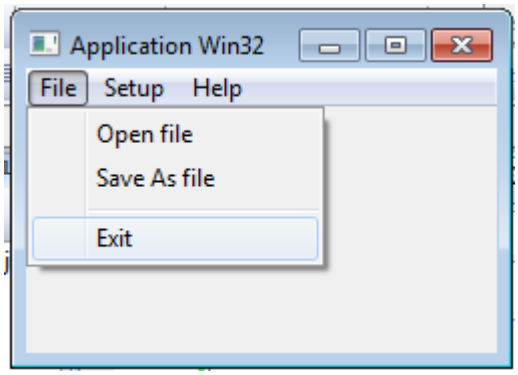
Диалоговое окно “ChooseColor” появляется в ответ на вызов функции ChooseColor. Приложения могут управлять начальным значением в этом диалоговом окне через указатель на структуру CHOOSECOLOR – единственный параметр, передаваемый функции ChooseColor (поле rgbResult этой структуры отвечает за выбранный цвет). Посредством этой структуры приложения могут также настраивать поведение диалогового окна, поддерживая обработчик

- С помощью диалогового “Font” окна можно выбрать шрифт, его размер, специальные эффекты, цвет текста.

Диалоговое окно “ChooseFont” появляется в ответ на вызов функции ChooseFont. Инициализируется оно через структуру CHOOSEFONT, которую также можно использовать для указания пользовательского обработчика и имени шаблона диалогового окна. Переменная lpLogFont этой структуры указывает на структуру LOGFONT, которая используется для инициализации диалогового окна и получения информации о новом выбранном шрифте при закрытии диалогового окна.

Задание:

- Требуется создать динамически подключаемое меню с выводом динамических окон:



Ход работы:

1. Описать глобальные переменные:

//Для шрифта

```
static char FontStyle[LF_FACESIZE];
static LOGFONT log_font;
static HFONT font;
CHOOSEFONT CF;
//Для цвета
static COLORREF rgb_text,CustColors[16];
	CHOOSECOLOR CC;

static char fullpath[256],filename[256],dir[256];
	OPENFILENAME ofn;
```

2. **В тело оконной функции в оператор обработки сообщения** WM_COMMAND: **включить следующий код:**

2.1. Для case ID_OPEN:

```
                                        {
                                        sprintf(fullpath,""); sprintf(filename,""); sprintf(dir,"");
	memset(&ofn,0,sizeof(OPENFILENAME));
	ofn.lStructSize=sizeof(OPENFILENAME);
	ofn.hwndOwner=hWnd;
	ofn.hInstance=hInst; // дескриптор копии приложения
	ofn.lpstrFilter="Текстовые файлы (*.txt)\0*.txt\0Все файлы (*.*)\0*.*\0";
	ofn.nFilterIndex=1;
	ofn.lpstrFile=fullpath;
	ofn.nMaxFile=sizeof(fullpath);
	ofn.lpstrFileTitle=filename;
	ofn.nMaxFileTitle=sizeof(filename);
	ofn.lpstrInitialDir=dir;
	ofn.lpstrTitle="Открыть в файле с именем...";
	ofn.Flags=OFN_PATHMUSTEXIST|OFN_OVERWRITEPROMPT|OFN_HIDEREADON
LY|OFN_EXPLORER;
	bool iCode=GetOpenFileName(&ofn);
		sprintf(dir,"%s",fullpath); dir[ofn.nFileOffset-1]='\0';
	if(iCode)
	{
		// пользователь выбрал файл, полное имя которого хранится в fullpath
	}
	break;}
```

2.2. Для case ID_SaveAs:

```
                                        {
                                        sprintf(fullpath,""); sprintf(filename,""); sprintf(dir,"");
	memset(&ofn,0,sizeof(OPENFILENAME));
	ofn.lStructSize=sizeof(OPENFILENAME);
	ofn.hwndOwner=hWnd;
	ofn.hInstance=hInst; // дескриптор копии приложения
	ofn.lpstrFilter="Текстовые файлы (*.txt)\0*.txt\0Все файлы (*.*)\0*.*\0";
	ofn.nFilterIndex=1;
	ofn.lpstrFile=fullpath;
	ofn.nMaxFile=sizeof(fullpath);
	ofn.lpstrFileTitle=filename;
```

```
ofn.nMaxFileTitle=sizeof(filename);
ofn.lpstrInitialDir=dir;
ofn.lpstrTitle="Сохранить в файле с именем...";
ofn.Flags=OFN_PATHMUSTEXIST|OFN_OVERWRITEPROMPT|OFN_HIDEREADON
LY|OFN_EXPLORER;
bool iCode=GetSaveFileName(&ofn);
        sprintf(dir,"%s",fullpath); dir[ofn.nFileOffset-1]='\0';
if(iCode)
{
        // пользователь выбрал файл, полное имя которого хранится в fullpath
}
break;}
```

2.3. Для case ID_COLOR:

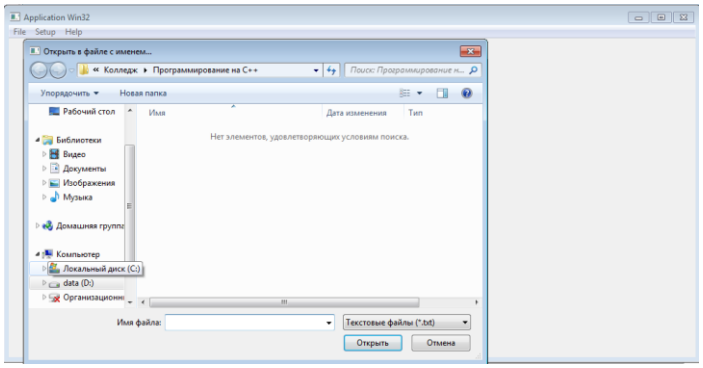
```
rgb_text=RGB(0,0,255);
for(int i=0; i<16; i++) CustColors[i]=0;
memset(&CC,0,sizeof(CHOOSECOLOR));
CC.lStructSize=sizeof(CHOOSECOLOR);
CC.hwndOwner=hWnd;
CC.hInstance=hWnd;
CC.lpCustColors=CustColors;
CC.Flags=CC_RGBINIT|CC_FULLOPEN;
CC.rgbResult=rgb_text;

if(ChooseColor(&CC))
{
        // пользователь выбрал цвет, информация о нем хранится в CC.rgbResult
        rgb_text=CC.rgbResult;
}
```

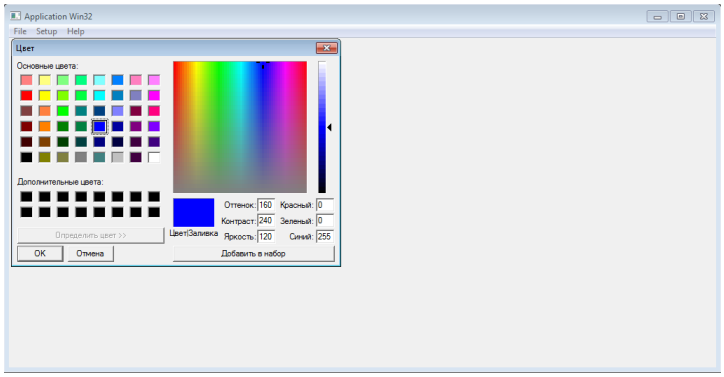
2.4. **Для** case ID_FONT:

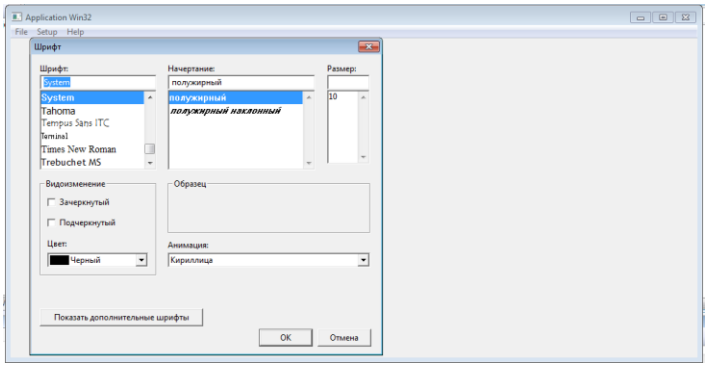
```
memset(&log_font,0,sizeof(LOGFONT));
log_font.lfWeight=FW_NORMAL;
log_font.lfCharSet=ANSI_CHARSET;
log_font.lfOutPrecision=OUT_DEFAULT_PRECIS;
log_font.lfClipPrecision=CLIP_DEFAULT_PRECIS;
log_font.lfQuality=DEFAULT_QUALITY;
log_font.lfPitchAndFamily=DEFAULT_PITCH|FF_MODERN;
sprintf(FontStyle,"System"); sprintf(log_font.lfFaceName,"System");
font=CreateFontIndirect(&log_font);
                                        memset(&CF,0,sizeof(CHOOSEFONT));
CF.lStructSize=sizeof(CHOOSEFONT);
CF.hwndOwner=hWnd;
CF.hInstance=hInst;
CF.lpLogFont=&log_font;
CF.Flags=CF_SCREENFONTS|CF_EFFECTS|CF_INITTOLOGFONTSTRUCT;
CF.rgbColors=RGB(0,0,0);
CF.lpszStyle=(LPSTR)FontStyle;
CF.nFontType=SCREEN_FONTTYPE;
CF.nSizeMin=4;
CF.nSizeMax=100;

if(ChooseFont(&CF))
```

```
{
        // пользователь выбрал шрифт, информация о нем хранится в log_font
        font=CreateFontIndirect(&log_font);
}
```

3. **Запустить программу и убедиться, что в главном окне появилось меню и для каждого пункта меню выводится соответствующее диалоговое окно.**







Лабораторная работа № 13

Тема: Тестирование элементов управления. Отладка модуля. Работа с кнопками и цветом

Цель: научиться использовать элементы управления: кнопка, список и изменять цвет прямоугольника.

Время выполнения: 80 минут

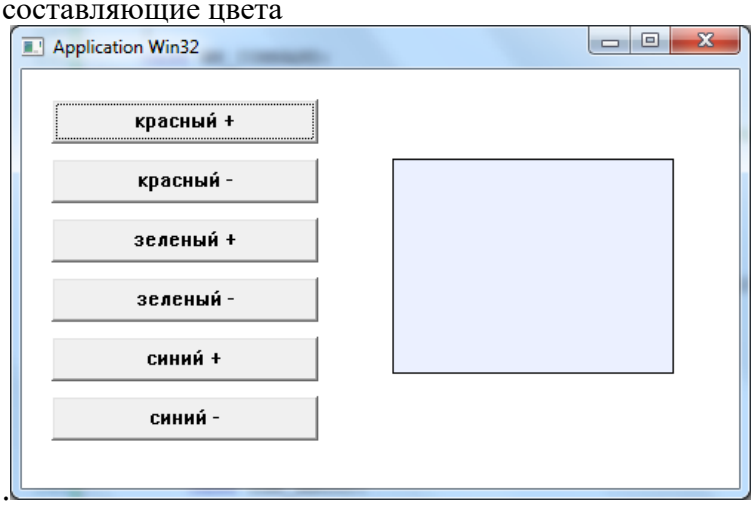
Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**

У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Постановка задачи:
Приложение позволяет изменять цвет прямоугольника, увеличивая или уменьшая составляющие цвета



Методические указания

Создание проекта Windows-приложения и Си-файла для его исходного кода:

- Создать проект Windows-приложения в Visual C++
- (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name -> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы из ПРИЛОЖЕНИЯ 1

- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

Порядок работы

Сначала создадим простейшее приложение Spisok.

В файл Spisok.h добавим определения констант-идентификаторов для элементов управления:

```
#define ID_Red1 1
#define ID_Red2 2
#define ID_Green1 3
#define ID_Green2 4
#define ID_Blue1 5
#define ID_Blue2 6
```

В файл Spisok.cpp добавим:

Глобальные переменные, соответствующие главному и дочерним окнам.

```
    HWND hRed1, hRed2, hGreen1,hGreen2,hBlue1,hBlue2 ;
```

В главной функции после оператора:

```
RegisterClass(&wc);
```

Вставить код:

```
          // Создаем главное окно приложения.
          hWnd = CreateWindow(ClassName, AppTitle, WS_OVERLAPPEDWINDOW,
   200, 200, 500, 320, NULL, NULL, hInstance, NULL);

  hRed1 = CreateWindow("button", "красный +",     WS_CHILD | WS_VISIBLE ,
   20, 20, 180, 30, hWnd, (HMENU) ID_Red1, hInstance, NULL);

  hRed2 = CreateWindow("button", "красный -",     WS_CHILD | WS_VISIBLE ,
   20, 60, 180, 30, hWnd, (HMENU) ID_Red2, hInstance, NULL);

  hGreen1 = CreateWindow("button", "зеленый +",     WS_CHILD | WS_VISIBLE ,
   20, 100, 180, 30, hWnd, (HMENU) ID_Green1, hInstance, NULL);

  hGreen2 = CreateWindow("button", "зеленый -",     WS_CHILD | WS_VISIBLE ,
   20, 140, 180, 30, hWnd, (HMENU) ID_Green2, hInstance, NULL);

  hBlue1 = CreateWindow("button", "синий +",     WS_CHILD | WS_VISIBLE ,
   20, 180, 180, 30, hWnd, (HMENU) ID_Blue1, hInstance, NULL);

  hBlue2= CreateWindow("button", "синий -",     WS_CHILD | WS_VISIBLE ,
   20, 220, 180, 30, hWnd, (HMENU) ID_Blue2, hInstance, NULL);

  if (!hWnd)
  {
     return FALSE;
  }

  ShowWindow(hWnd, nCmdShow);
  UpdateWindow(hWnd);
```

Изменения в оконной процедуре:

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
      int wmId, wmEvent;
      PAINTSTRUCT ps;
```

```
      HDC hdc;

      static short Red=255,Green=255,Blue=255;// составляющие цвета
      static COLORREF mycolor=RGB(Red,Green,Blue);//текущий цвет
      HBRUSH hBr;//кисть
      RECT Rect;//прямоугольник для окрашивания
      Rect.left=250;
      Rect.top=60;
      Rect.right=440;
      Rect.bottom=205;

      switch (message)
      {
      case WM_COMMAND:
            wmId    = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            // Разобрать выбор в меню или сообщение от дочернего окна
            switch (wmId)
      {     case ID_Red1: Red+=5;if(Red>255)Red=255;
                  break;
            case ID_Red2: Red-=5;if(Red<0)Red=0;
                  break;
            case ID_Green1: Green+=5;if(Green>255)Green=255;
                  break;
            case ID_Green2: Green-=5;if(Green<0)Green=0;
                  break;
            case ID_Blue1: Blue+=5;if(Blue>255)Blue=255;
                  break;
            case ID_Blue2: Blue-=5;if(Blue<0)Blue=0;
                  break;
            case IDM_EXIT:
                  DestroyWindow(hWnd);
                  break;
            }
      // создадим цвет
         mycolor=RGB(Red,Green,Blue);
      // помечаем область, соответствующую прямоугольнику, требующей обновления
       InvalidateRect(hWnd,&Rect,TRUE);
       UpdateWindow(hWnd);
      break;


      case WM_PAINT:
//получаем контекст для рисования
            hdc = BeginPaint(hWnd, &ps);
//создаем сплошную кисть нужного цвета
            hBr=CreateSolidBrush(mycolor);
//окрашиваем прямоугольник Rect кистью hBr
            FillRect(hdc,&Rect,hBr);
//создаем сплошную кисть черного цвета
            hBr=CreateSolidBrush(RGB(0,0,0));
//рисуем рамку вокруг прямоугольника
            FrameRect(hdc,&Rect,hBr);
//заканчиваем рисовать
            EndPaint(hWnd, &ps);
            break;

      case WM_DESTROY:
            PostQuitMessage(0);
            break;
      default:
            return DefWindowProc(hWnd, message, wParam, lParam);
      }
```

```
    return 0;
}
```

Ответить на вопросы:

Какие сообщения обрабатываются в данном приложении?
Какие типы переменных используются в процедуре WndProc?
Какие функции библиотеки WIN32 API используются в данном приложениии?

Задание

1. При каждом изменении цвета выводить значения его составляющих.
2. Вывод осуществлять в элемент управления Static.
3. Вывод осуществлять в элемент редактирования Edit.
4. Вывод осуществлять при помощи функции TextOut.

Лабораторная работа № 14

Тема: Тестирование элементов управления. Отладка модуля. Работа со списками

Цель: научиться использовать три элемента управления: кнопка, список и выпадающий список.

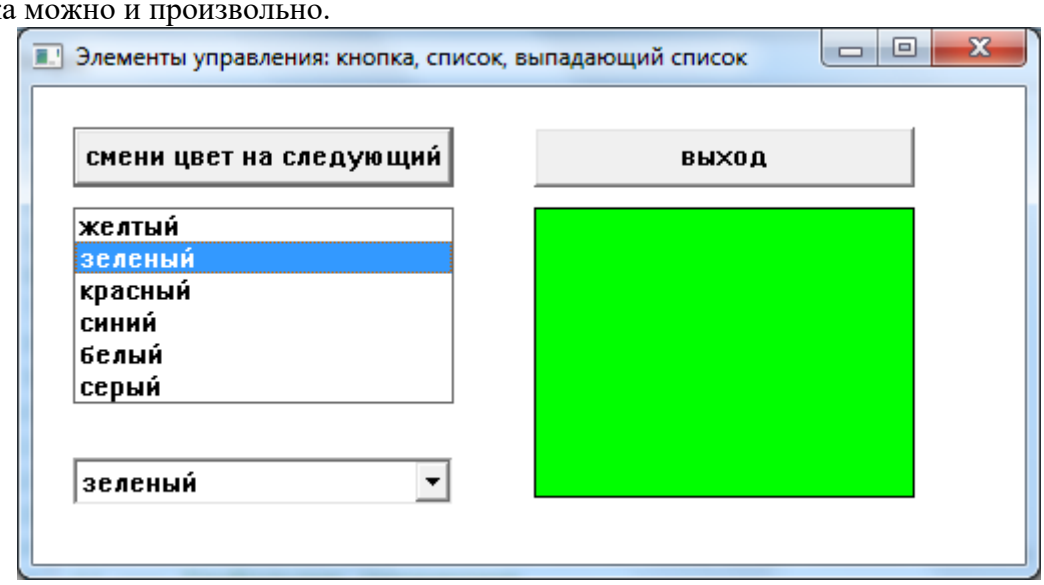
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Постановка задачи:
В списках содержится одинаковый набор цветов, в которые можно окрасить прямоугольную область. При нажатии кнопки «Смени цвет на следующий» в каждом списке выбирается очередной элемент, и окрашивается прямоугольник. При нажатии на кнопку выбор цвета осуществляется циклически, то есть после серого будет выбран желтый цвет. Выбирать цвет из списка можно и произвольно.



Методические указания
Создание проекта Windows-приложения и Си-файла для его исходного кода:

- Создать проект Windows-приложения в Visual C++
- (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name -> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы из ПРИЛОЖЕНИЯ 1
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

Порядок работы

Сначала создадим простейшее приложение Spisok.
В файл Spisok.h добавим определения констант-идентификаторов для элементов управления:

```
#define ID_ButtonSelect 1
#define ID_List 2
#define ID_ButtonExit 3
#define ID_Combo 4
#define MAX_LOADSTRING 100
```

В файл Spisok.cpp добавим:
1. Глобальные переменные, соответствующие главному и дочерним окнам.

```
HWND hButtonSelect, hList, hButtonExit, hCombo;
```

2. В главной функции после оператора:

```
    // Обновляем содержимое клиентской области окна.
    UpdateWindow(hWnd);
```

добавим блок создания элементов управления:

```
hButtonSelect = CreateWindow("button", "смени цвет на следующий", WS_CHILD|
BS_DEFPUSHBUTTON| WS_VISIBLE, 20,20,190,30, hWnd,(HMENU)ID_ButtonSelect,
hInstance, NULL);

hList = CreateWindow("listbox", NULL, WS_CHILD| WS_BORDER| WS_VISIBLE|LBS_NOTIFY
, 20,60,190,100, hWnd,(HMENU)ID_List, hInstance, NULL);

hButtonExit = CreateWindow("button", "выход", WS_CHILD| BS_PUSHBUTTON|
WS_VISIBLE, 250,20,190,30, hWnd,(HMENU)ID_ButtonExit, hInstance, NULL);

hCombo = CreateWindow("combobox", NULL, WS_CHILD|  WS_BORDER|
WS_VISIBLE|CBS_SIMPLE |CBS_DROPDOWNLIST ,20,185,190,160, hWnd,(HMENU)ID_Combo,
hInstance, NULL);

//  заполним списки названиями цветов, хранящихся в массиве Color
  char Color[6][8]={"желтый","зеленый","красный","синий","белый","серый"};
  int i;
  for(i=0;i<6;i++)
  {SendMessage(hList,LB_ADDSTRING,0,(LPARAM)Color[i]);
   SendMessage(hCombo,CB_ADDSTRING,0,(LPARAM)Color[i]);
   }
```

Изменения в оконной процедуре:

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
      int wmId, wmEvent;
      PAINTSTRUCT ps;
      HDC hdc;
int nSelect=0;//номер выбранной строки
      HRGN hReg=CreateRectRgn(250,60,440,205);//прямоугольная область для
окрашивания
      HBRUSH hBr;//кисть
      static COLORREF mycolor;//текущий цвет
      //набор цветов
static COLORREF color_set[6]={RGB(255,236,128), RGB(0,255,0), RGB(255,0,0),
RGB(0,0,255), RGB(255,255,255), RGB(118,118,118)};

      switch (message)
      {
      case WM_COMMAND://  какой элемент управления послал сообщение
      switch (LOWORD(wParam))
      {
// нажата кнопка смены цвета
      case ID_ButtonSelect:
      // посылаем сообщение списку и узнаем, какая в нем выбрана строка
      //увеличиваем номер и сохраняем в nSelect
       nSelect  = SendMessage(hList, LB_GETCURSEL, 0,0L)+1;
      // если была выбрана последняя строка, то следующей будет первая
       if(nSelect>=SendMessage(hList, LB_GETCOUNT, 0,0L)) nSelect=0;
      //посылаем сообщения спискам, чтобы выбрать в них строку nSelect
       SendMessage(hList, LB_SETCURSEL, (WPARAM)nSelect,0L);
       SendMessage(hCombo,CB_SETCURSEL, (WPARAM)nSelect,0L);
       break;
// пришло сообщение от списка
      case ID_List:
      // выясняем, какую строку выбрали
      nSelect  = SendMessage(hList, LB_GETCURSEL, 0,0L);
      // устанавливаем тот же выбор в выпадающем списке
      SendMessage(hCombo,CB_SETCURSEL, (WPARAM)nSelect,0L);
      break;
// пришло сообщение от выпадающего списка
```

```
      case ID_Combo:  nSelect  = SendMessage(hCombo, CB_GETCURSEL, 0,0L);
      SendMessage(hList, LB_SETCURSEL, (WPARAM)nSelect,0L);
      break;
// пришло сообщение от кнопки Выход
      case ID_ButtonExit:
      // посылаем сообщение главному окну о завершении работы
      SendMessage(hWnd,WM_DESTROY , 0,0L);
      break;
      }

// определяем, какой цвет выбрали
 mycolor=color_set[nSelect];
// помечаем главное окно как целиком требующее обновления
 InvalidateRect(hWnd,0,TRUE);
// обновляем окно, ему придет сообщение WM_PAINT
 UpdateWindow(hWnd);
 break;


//это сообщение будет приходить каждый раз, когда окно будет изменять свое
состояние с активного на неактивное
      case WM_ACTIVATE:
            nSelect=0;
            SendMessage(hCombo,CB_SETCURSEL, (WPARAM)nSelect,0L);
            SendMessage(hList, LB_SETCURSEL, (WPARAM)nSelect,0L);
            mycolor=color_set[nSelect];
            InvalidateRect(hWnd,0,TRUE);
            UpdateWindow(hWnd);
            break;
//сообщение о перерисовке содержимого окна
      case WM_PAINT:
//получаем контекст для рисования
            hdc = BeginPaint(hWnd, &ps);
//создаем сплошную кисть нужного цвета
            hBr=CreateSolidBrush(mycolor);
            //окрашиваем область hReg кистью hBr
FillRgn(hdc,hReg,hBr);
//создаем сплошную кисть черного цвета
            hBr=CreateSolidBrush(RGB(0,0,0));
            //рисуем рамку вокруг области
            FrameRgn(hdc,hReg,hBr,1,1);
            //заканчиваем рисовать
            EndPaint(hWnd, &ps);
            break;
//сообщение о завершении работы
      case WM_DESTROY:
            //послать сообщение WM_QUIT
            PostQuitMessage(0);
            break;
//все остльные сообщения будут обработаны по умолчаню
      default:
            return DefWindowProc(hWnd, message, wParam, lParam);
      }
      return 0;
}
```

Задание

При каждом изменении цвета выводить значения его составляющих.

Вывод осуществ**влять в Edit**

Вывод осуществлять при помощи функции TextOut

Лабораторная работа № 15

Тема: Работа с элементом редактирования – заголовок окна

Цель: научиться использовать элементы управления: кнопка, редактор, обрабатывать текст из редактора.

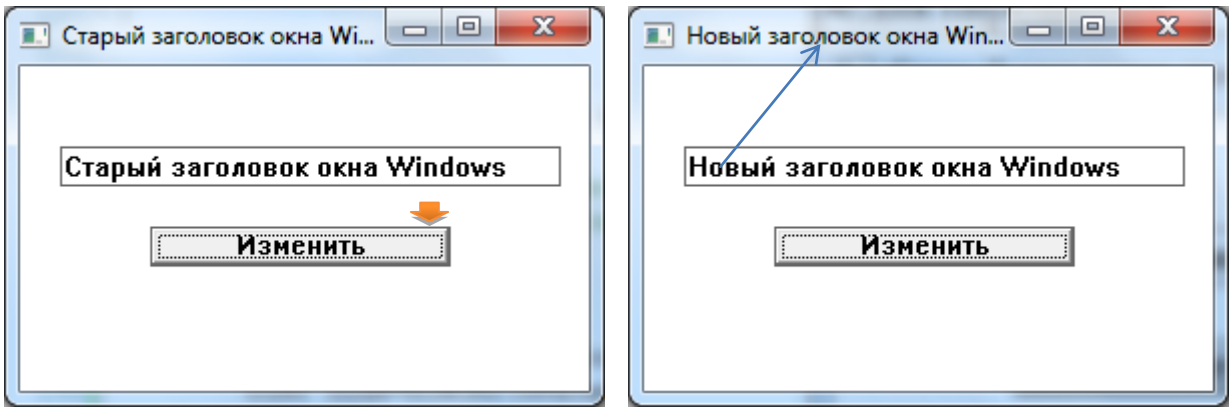
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Постановка задачи: В этом приложении используются два элемента управления – кнопка и элемент редактирования. По нажатию кнопки заголовок окна изменяется на текст, который введен в элемент редактирования.



Методические указания

Создание проекта Windows-приложения и Си-файла для его исходного кода:

- Создать проект Windows-приложения в Visual C++
- (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name -> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы из ПРИЛОЖЕНИЯ 1
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

Порядок работы

Сначала создадим простейшее приложение Tazk

В файл Tazk.h добавим определения констант-идентификаторов для элементов управления:

```
#define ID_Button 1
#define ID_Edit 2
```

В файл Tazk.cpp добавим:

Глобальные переменные, соответствующие главному и дочерним окнам.

```
HWND hButton, hEdit;
```

В главной функции после оператора:

```
RegisterClass(&wc);
```

Вставить код:

```
  hWnd = CreateWindow(ClassName, AppTitle, WS_OVERLAPPEDWINDOW| WS_VISIBLE,
      200, 200, 300, 200, NULL, NULL, hInstance, NULL);

//Создание кнопки
  hButton= CreateWindow("button", "Изменить",
  WS_CHILD | WS_VISIBLE | BS_DEFPUSHBUTTON,
  65,80,//x и y
  150, 20,//ширина и высота
  hWnd,// родитель
  (HMENU)ID_Button,// идентификатор кнопки задать самим
  hInstance, NULL);
//Создание элемента редактирования
  hEdit = CreateWindow("edit", "Новый заголовок ",
   WS_CHILD | WS_VISIBLE | WS_BORDER |ES_LEFT,
  20,40,//x и y(координаты левого верхнего угла)
  250, 20,//ширина и высота
  hWnd,// родитель
  (HMENU)ID_Edit,// идентификатор edit'a задать самим
```

```
  hInstance, NULL);
// Главное окно создавалось невидимым, отображаем его
  ShowWindow(hWnd, nCmdShow);
  UpdateWindow(hWnd);
```

Изменения в оконной процедуре:

```
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
     char  chText[70];// массив символов для хранения строки
     UINT cbCount=0;// число символов в строке
   switch(msg)
      {
            case WM_COMMAND:
      {     UINT idCtl=LOWORD(wParam); // идентификатордочернегоокна
            UINT code=HIWORD(wParam);  // кодуведомления
            HWND hChild=(HWND)lParam;  // дескриптордочернегоокна

            if(idCtl==ID_Button&&code==BN_CLICKED)
            {
                        // Получаем от редактора текста содержимое.
         // Функция возвращает количество прочитанных байт.
         cbCount = SendMessageA(hEdit, EM_GETLINE, 0, (LPARAM)(LPSTR)chText);
         // Строка должна оканчиваться двоичным нулем
         chText[cbCount] = '\0';
         //Изменяем заголовок главного окна
         // Ко второму параметру применяем явное преобразование типа
         SetWindowText(hWnd,(LPSTR)chText );


               }
      }; return 0;


            // Пользовательудалилокно.
            case WM_DESTROY:
            {
                  // Если данная функция является оконной функцией главного
окна, то
                  // следует в очередь сообщений приложения послать сообщение
WM_QUIT
                  PostQuitMessage(0);
            }; break;

            // Необработанные сообщения передаем в стандартную
            // функцию обработки сообщений по умолчанию.
            default: return DefWindowProc(hWnd, msg, wParam, lParam);
      }
      return 0l;
```

} **Ответить на вопросы:**

Какие сообщения обрабатываются в данном приложении?

Какие типы переменных используются в процедуре WndProc?

Какие функции библиотеки WIN32 API используются в данном приложениии?

Задание

1. Добавить еще один элемент редактирования и отображать в нем старый заголовок окна.
2. Указание: перед сменой заголовка окна сохранять старый заголовок и посылать его второму элементу редактирования.

Лабораторная работа № 16

Тема: Чтение данных из файла

Цель: научиться использовать элементы управления: кнопка, статический элемент, обрабатывать текст из редактора.

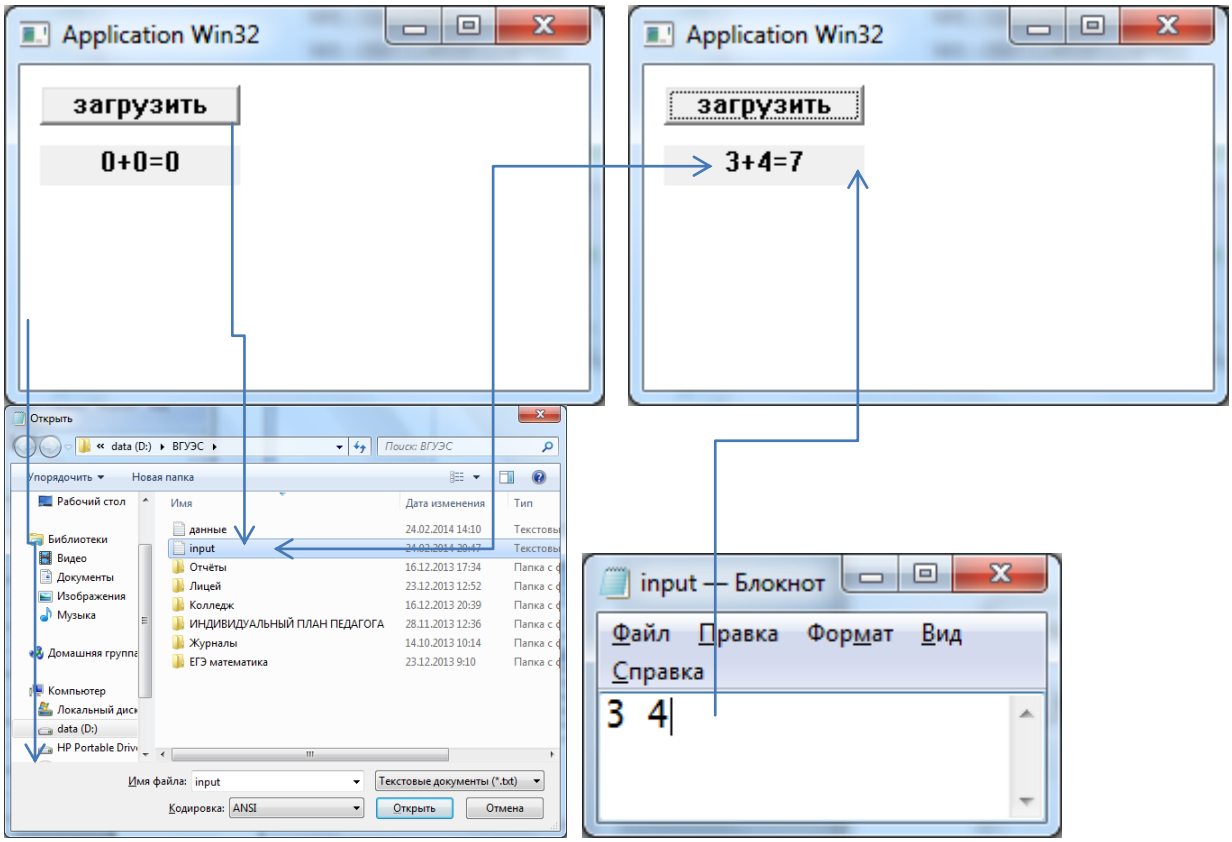
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Постановка задачи: В этом приложении используются два элемента управления – кнопка и статический текст. Данные хранятся в текстовом файле и представляют собой два целых числа. После прочтения чисел из файла на экран выводится сумма этих чисел.



Методические указания

Создание проекта Windows-приложения и Си-файла для его исходного кода:

- Создать проект Windows-приложения в Visual C++
- (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name -> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы из ПРИЛОЖЕНИЯ 1
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

Чтобы использовать стандартный диалог, подключается файл заголовков
#include <commdlg.h> и
используется тип данных - структура, описывающая вид диалога
OPENFILENAME ofn.

Имя файла хранится в переменной file_name[100]. После нажатия на кнопку «загрузить» открывается окно, позволяющее выбрать файл с исходными данными «input.txt», созданного приложением «Блокнот».
Обрабатывается событие `IDB_Load`.

Порядок работы

Сначала создадим простейшее приложение Tazk
В файл Tazk.h добавим определения констант-идентификаторов для элементов управления:

Для того чтобы использовать стандартный диалог, нужно подключить файл заголовков
```
#include <commdlg.h>
```
Для того чтобы использовать функции работы с файлами, нужно подключить файл заголовков
```
#include <stdio.h>
```
Определим идентификатор кнопки «загрузить»:
```
#define IDB_Load 1
```

В файл Tazk.cpp добавим:
Описать глобальные переменные, соответствующие главному и дочерним окнам.
```
HWND hButton, hStatic;
```
В главной функции после оператора:
```
RegisterClass(&wc);
```
вставить код:

```
hButton = CreateWindow("button", "загрузить" ,
WS_CHILD|WS_VISIBLE| BS_PUSHBUTTON, 10, 10, 100, 20, hWnd,
(HMENU)IDB_Load, hInstance, NULL);
hStatic = CreateWindow("Static", "0+0=0" , WS_CHILD|WS_VISIBLE|
SS_CENTER, 10, 40, 100, 20, hWnd, NULL, hInstance, NULL);
```

Изменения в оконной процедуре:
```
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{    int wmId, wmEvent;
     PAINTSTRUCT ps;
     HDC hdc;
     char file_name[100]="";// имя файла
     OPENFILENAME ofn;// структура, описывающая вид диалога
     switch (message)
     {
     case WM_COMMAND:
          wmId    = LOWORD(wParam);
          wmEvent = HIWORD(wParam);
          // Разобрать выбор в меню:
          switch (wmId)
          {
          case IDM_EXIT:
          DestroyWindow(hWnd);
          break;
case IDB_Load:
memset(&ofn, 0, sizeof(OPENFILENAME));// заполнить нулями
структуру ofn
       //заполним только основные поля
```

```
       ofn.lStructSize=sizeof (OPENFILENAME);// размер структуры

ofn.Flags=OFN_FILEMUSTEXIST|OFN_PATHMUSTEXIST|OFN_HIDEREADONLY;//
внешний вид
       ofn.hwndOwner=hWnd;// идентификатор окна, создавшего
диалоговое окно
       ofn.lpstrDefExt="txt";
       ofn.lpstrFile= file_name;// адрес строки,где будет
содержаться имя выбранного файла, самое нужное поле
       ofn.lpstrFilter="Text Files\0*.txt;Any Files\0*.*\0\0";
       ofn.lpstrInitialDir="C:\\";// с какой папки начать просмотр
       ofn.lpstrTitle="Загрузить данные из файла:";//заголовок
диалога
       ofn.nFilterIndex=1;
       ofn.nMaxFile=50;// размер буфера для записи пути к файлу

      if (GetOpenFileNameA(&ofn))// вызвать стандартный диалог
     // если работа с диалогом завершилась успешно
       { FILE* f=fopen(ofn.lpstrFile,"r");
          int nA,nB;
          char str[10];
          fscanf(f,"%d %d",&nA,&nB);
           sprintf(str,"%d*%d=%d",nA,nB,nA*nB);
          SetWindowText( hStatic, str); // выведем информацию в
окно
       }
       // если была нажата кнопка «Отмена»
     else MessageBoxA(hWnd, "не выбран файл"," Сообщение " ,
MB_OK);
    }
          break;
     case WM_PAINT:
          hdc = BeginPaint(hWnd, &ps);
          // TODO: добавьте любой код отрисовки...
          EndPaint(hWnd, &ps);
          break;
     case WM_DESTROY:
          PostQuitMessage(0);
          break;
     default:
          return DefWindowProc(hWnd, message, wParam, lParam);
     }
     return 0;
}
```

Ответить на вопросы:

Какие сообщения обрабатываются в данном приложении?

Какие типы переменных используются в процедуре WndProc?

Какие функции библиотеки WIN32 API используются в данном приложениии?

Какая функция вызывает стандартный диалог?

Какая структура используется для обращения к файлу с входными данными?

Какие переменные содержат входные данные?
Сделать блок-схему процедуры WndProc.

Задание

3. Добавить еще один элемент редактирования и отображать в нем старый заголовок окна.
4. Указание: перед сменой заголовка окна сохранять старый заголовок и посылать его второму элементу редактирования.

Лабораторная работа № 17

Тема: Чтение данных (два числа) из файла и сохранение текстовой строки в другой файл.

Цель: научиться использовать элементы управления: кнопка, статический элемент, обрабатывать текст из редактора, создавать диалог выбора файла, чтение и запись данных в файл и из файла.

Время выполнения: 80 минут

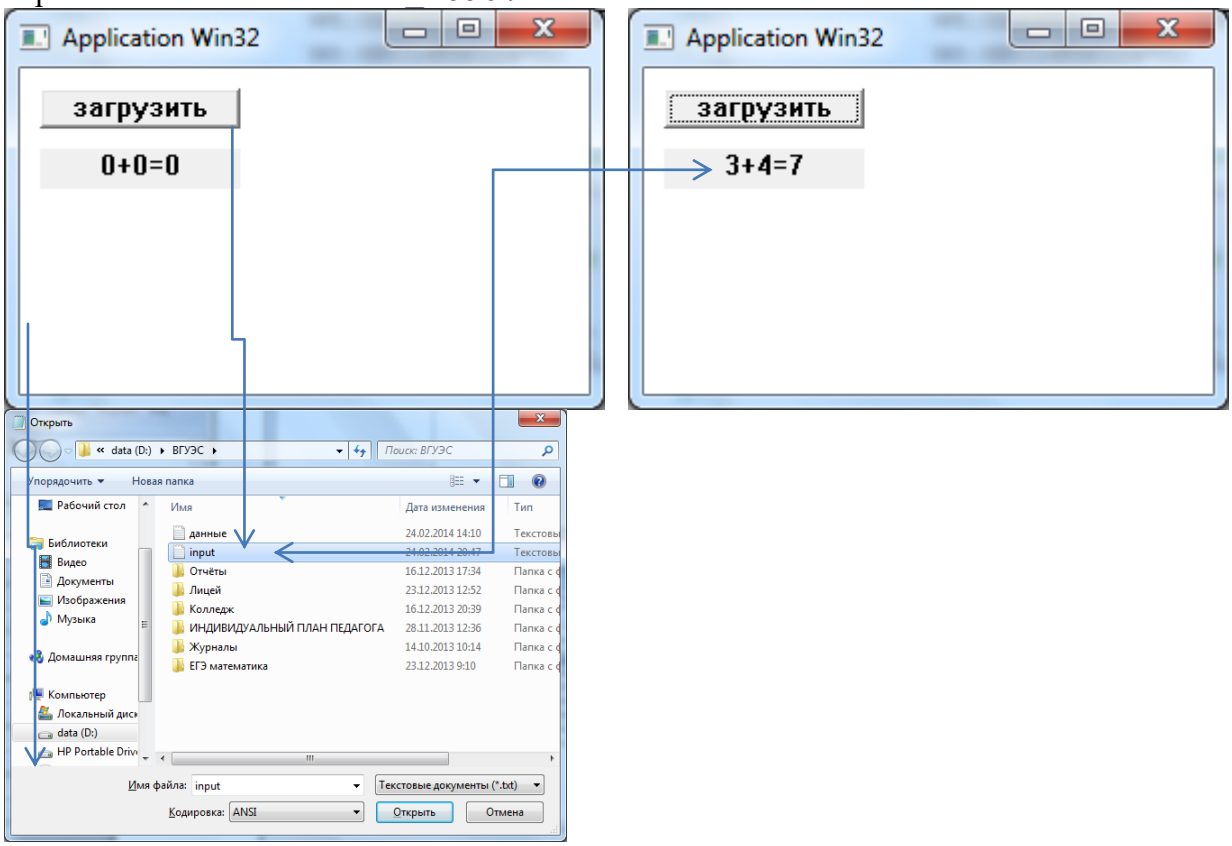
Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Методические указания: В этом приложении используются два элемента управления – кнопка и статический текст. Данные хранятся в текстовом файле и представляют собой два

целых числа. После прочтения чисел из файла на экран выводится сумма этих чисел, а в выходной файл копируется текстовая строка.
Функции и структуры, которые необходимы для использования диалоговых окон общего пользования, определяются в файле commdlg.h. Чтобы использовать стандартный диалог, подключается файл заголовков
#include <commdlg.h> и
используется тип данных - структура, описывающая вид диалога OPENFILENAME ofn.
Диалоговое окно “Open” открывается при вызове функции GetOpenFileName. Единственным параметром функции является указатель на структуру OPENFILENAME.
Диалоговое окно “Save As” отображается в ответ на вызов функции GetSaveFileName.
Имя файла хранится в переменной file_name[100]. После нажатия на кнопку «загрузить» открывается окно, позволяющее выбрать файл с исходными данными «input.txt», созданного приложением «Блокнот». В файле хранятся два числа, которые должны быть прочитаны и обработаны. Результат обработки входных данных записывается в статический элемент Static и в текстовый файл.
Обрабатывается событие `IDB_Load`.



Ход работы:
Создание проекта Windows-приложения и Си-файла для его исходного кода:

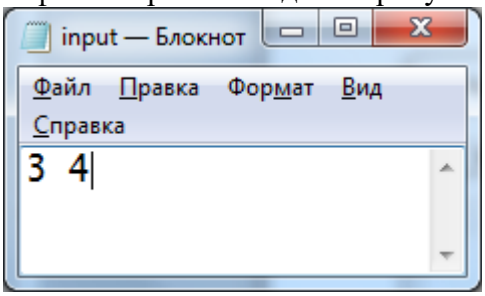
- Создать проект Windows-приложения в Visual C++
- (последовательность действий*: выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 Project -> ввести имя проекта в поле Project name -> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK*).

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы из ПРИЛОЖЕНИЯ 1
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;

- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

1. В общей папке студентов создать текстовые файлы input.txt и output.txt с помощью приложения Блокнот.
   В файле input.txt создать строку из двух чисел через один пробел, например: 3 4



2. Создать простейшее приложение Tazk, которое необходимо скопировать из ПРИЛОЖЕНИЯ 1.
3. В файл Tazk.h добавить определения констант-идентификаторов для элементов управления:

Для того чтобы использовать стандартный диалог, нужно подключить файл заголовков
```
#include <commdlg.h>
```
Для того чтобы использовать функции работы с файлами, нужно подключить файл заголовков
```
#include <stdio.h>
```
Определить идентификатор кнопки «загрузить»:
```
#define IDB_Load 1
```

Изменения в оконной процедуре WndProc:
После обращения к функции:
```
SetWindowText( hStatic, str); // выведем информацию в окно
```

вставить код для создания диалога выбора файла с выходными данными и сохранения строки str в файле с помощью функции `fprintf(fw,"%s",str);`

```
ofn.lpstrTitle="Сохранить данные в файл:";//заголовок диалога
if (GetSaveFileName(&ofn))// вызвать стандартный диалог
    {
  FILE* fw=fopen(ofn.lpstrFile,"w+");
  fprintf(fw,"%s",str);
    }
```

Ответить на вопросы:
Какие сообщения обрабатываются в данном приложении?
Какие типы переменных используются в процедуре WndProc?
Какие функции библиотеки WIN32 API используются в данном приложении?
Какая функция считывает входные данные из файла?
Какая функция сохраняет выходные данные в файл?
Какая функция вызывает стандартный диалог?
Какая структура используется для диалога обращения к файлам с входными и выходными данными?
Какие переменные содержат входные данные?

Сделать блок-схему процедуры WndProc.

Задание

Добавить еще два элемента редактирования и отображать в них имена входного и выходного файлов.

```
static char fullpath[256],filename[256],dir[256];
        OPENFILENAME ofn;
        ...
        sprintf(fullpath,""); sprintf(filename,""); sprintf(dir,"");
        ...
        memset(&ofn,0,sizeof(OPENFILENAME));
        ofn.lStructSize=sizeof(OPENFILENAME);
        ofn.hwndOwner=hWnd;
        ofn.hInstance=hInst; // дескриптор копии приложения
        ofn.lpstrFilter="Текстовые файлы (*.txt)\0*.txt\0Все файлы
(*.*)\0*.*\0";
        ofn.nFilterIndex=1;
        ofn.lpstrFile=fullpath;
        ofn.nMaxFile=sizeof(fullpath);
        ofn.lpstrFileTitle=filename;
        ofn.nMaxFileTitle=sizeof(filename);
        ofn.lpstrInitialDir=dir;
        ofn.lpstrTitle="Сохранить в файле с именем...";
        ofn.Flags=OFN_PATHMUSTEXIST|OFN_OVERWRITEPROMPT|OFN_HIDEREADONLY|OFN_EXP
LORER;
        int iCode=GetSaveFileName(&ofn);
        sprintf(dir,"%s",fullpath); dir[ofn.nFileOffset-1]='\0';

        if(iCode)
        {
                // пользователь выбрал файл, полное имя которого хранится в
fullpath
        }
```

ПРИЛОЖЕНИЕ № 1

```
#include "stdafx.h"
#include <math.h>
#include <sstream>
#include <iostream>
#include <string>
#include <windows.h>
// --- Описание функции главного окна
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam);

// --- Глобальные переменные
HINSTANCE hInst;                                        // Дескриптор
экземпляра приложения
char ClassName[]="Window";                              // Название класса окна
char AppTitle[]="Application Win32";            // Заголовок главного окна
 HWND hButton, hStatic;
// --- Функция WinMain
int WINAPI WinMain(
            HINSTANCE hInstance,                        // Дескриптор экземпляра
приложения
            HINSTANCE hPrevInstance,            // В Win32 всегда равен NULL
            LPSTR lpCmdLine,                            // Указатель на командную
строку. Он позволяет
                                                                                //
приложению получать данные из командной строки.
```

```
            int nCmdShow                                    // Определяет, как
приложение первоначально
                                                                        //
отображается на дисплее: пиктограммой
                                                                        //
(nCmdShow = SW_SHOWMINNOACTIVE)
                                                                        //
или в виде открытого окна

      //(nCmdShow = SW_SHOWNORMAL).
            )
{
            WNDCLASS wc;                  // Структура для информации о класса
окна
            HWND hWnd;        // Дескриптор главного окна приложения
            MSG msg;          // Структура для хранения сообщения

            // Сохраняем дескриптор экземпляра приложения в глобальной
переменной,
            // чтобы при необходимости воспользоваться им в функции окна.
            hInst=hInstance;

            // --- Проверяем, было ли приложение запущено ранее.
            // Воспользуемся функцией FindWindow, которая позволяет найти окно
верхнего
            // уровня по имени класса или по заголовку окна:
            //          HWND FindWindow(LPCTSTR lpClassName, LPCTSTR
lpWindowName);
            // Через параметр lpClassName передается указатель на текстовую
строку, в которую
            // необходимо записать имя класса искомого окна. На базе одного и
того же класса
            // можно создать несколько окон. Если необходимо найти окно с
заданным заголовком,
            // то имя заголовка следует передать через параметр lpWindowName.
Если же подойдет
            // любое окно, то параметр lpWindowName может иметь значение NULL.
            if((hWnd=FindWindow(ClassName, NULL))!=NULL)
            {
                        // Пользователь может не помнить, какие приложения
уже запущены,
                        // а какие нет. Когда он запускает приложение, то
ожидает, что на экране
                        // появится его главное окно. Поэтому, если
приложение было запущено
                        // ранее, целесообразно активизировать и выдвинуть
на передний план
                        // его главное окно. Это именно то, к чему
приготовился пользователь.
                        if(IsIconic(hWnd)) ShowWindow(hWnd, SW_RESTORE);
                        SetForegroundWindow(hWnd);

                        // Найдена работающая копия - работа новой копии
прекращается.
                        return FALSE;
            }

            // --- Работающая копия не найдена - функция WinMain приступает к
инициализации.
            // Заполнение структуры WNDCLASS для регистрации класса окна.
            memset(&wc, 0, sizeof(wc));
            wc.lpszClassName=ClassName;
      // Имя класса окон
```

```
            wc.lpfnWndProc=(WNDPROC)WndProc;
      // Адрес оконной функции
            wc.style=CS_HREDRAW|CS_VREDRAW;
      // Стиль класса окон
            wc.hInstance=hInstance;
      // Экземпляр приложения
            wc.hIcon=LoadIcon(NULL,IDI_APPLICATION);                              //
Пиктограмма для окон
            wc.hCursor=LoadCursor(NULL,IDC_ARROW);
      // Курсор мыши для окон
            wc.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH);             //
Кисть для окон
            wc.lpszMenuName=NULL;
      // Ресурс меню окон
            wc.cbClsExtra=0;
      // Дополнительная память
            wc.cbWndExtra=0;
      // Дополнительная память

            // Регистрация класса окна.
            RegisterClass(&wc);

            // Создаем главное окно приложения.
            hWnd=CreateWindow(
                              ClassName,
      // Имя класса окон
                              AppTitle,
      // Заголовок окна
                              WS_OVERLAPPEDWINDOW,                        // Стиль
окна
                              400, 200, 400, 200,
                              NULL,
      // Дескриптор окна-родителя
                              NULL,
      // Дескриптор меню окна
                              hInst,
      // Дескриптор экземпляра приложения
                              NULL);
      // Дополнительная информация
            if(!hWnd)
            {
                              // Окно не создано, выдаем предупреждение.
                              MessageBox(NULL,"Create:
error",AppTitle,MB_OK|MB_ICONSTOP);
                              return FALSE;
            }

            // Отображаем окно.
            ShowWindow(hWnd, nCmdShow);

            // Обновляем содержимое клиентской области окна.
            UpdateWindow(hWnd);
 hButton = CreateWindow("button", "загрузить данные из файла" ,
WS_CHILD|WS_VISIBLE| BS_PUSHBUTTON, 10, 10, 200, 20, hWnd, (HMENU)IDB_Load,
hInstance, NULL);
hStatic = CreateWindow("Static", "0+0=0" , WS_CHILD|WS_VISIBLE| SS_CENTER, 10,
40, 100, 20, hWnd, NULL, hInstance, NULL);

            // Запускаем цикл обработки очереди сообщений. Функция GetMessage
получает
            // сообщение из очереди, выдает false при выборке из очереди
сообщения WM_QUIT
            while(GetMessage(&msg, NULL, 0, 0))
```

```
            {
                                // Преобразование некоторых сообщений, полученных
с помощью клавиатуры
                                TranslateMessage(&msg);

                                // Отправляем сообщение оконной процедуре
                                DispatchMessage(&msg);
            }

            return msg.wParam;
}

// --- Функция окна
LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{     int wmId, wmEvent;
      PAINTSTRUCT ps;
      HDC hdc;
      char file_name[100]="";// имя файла
      OPENFILENAME ofn;// структура, описывающая вид диалога
      switch (message)
      {
      case WM_COMMAND:
            wmId    = LOWORD(wParam);
            wmEvent = HIWORD(wParam);
            // Разобрать выбор в меню:
            switch (wmId)
            {
            case IDM_EXIT:
            DestroyWindow(hWnd);
            break;
case IDB_Load:
memset(&ofn, 0, sizeof(OPENFILENAME));// заполнить нулями структуру ofn
        //заполним только основные поля
        ofn.lStructSize=sizeof (OPENFILENAME);// размер структуры
        ofn.Flags=OFN_FILEMUSTEXIST|OFN_PATHMUSTEXIST|OFN_HIDEREADONLY;//
внешний вид
        ofn.hwndOwner=hWnd;// идентификатор окна, создавшего диалоговое окно
        ofn.lpstrDefExt="txt";
        ofn.lpstrFile= file_name;// адрес строки,где будет содержаться имя
выбранного файла, самое нужное поле
        ofn.lpstrFilter="Text Files\0*.txt;Any Files\0*.*\0\0";
        ofn.lpstrInitialDir="C:\\";// с какой папки начать просмотр
        ofn.lpstrTitle="Загрузить данные из файла:";//заголовок диалога
        ofn.nFilterIndex=1;
        ofn.nMaxFile=50;// размер буфера для записи пути к файлу

       if (GetOpenFileNameA(&ofn))// вызвать стандартный диалог
      // если работа с диалогом завершилась успешно
        { FILE* f=fopen(ofn.lpstrFile,"r");
            int nA,nB;
            char str[20];
            fscanf(f,"%d %d",&nA,&nB);//ввод исходных данных из файла с именем f
в переменные nA,nB.
           sprintf(str,"%d*%d=%d",nA,nB,nA*nB);
            SetWindowText( hStatic, str); // выведем информацию в окно

      }
      // если была нажата кнопка «Отмена»
      else MessageBoxA(hWnd, "не выбран файл"," Сообщение " , MB_OK);
    }
            break;
      case WM_PAINT:
            hdc = BeginPaint(hWnd, &ps);
```

```
            // TODO: добавьте любой код отрисовки...
            EndPaint(hWnd, &ps);
            break;
      case WM_DESTROY:
            PostQuitMessage(0);
            break;
      default:
            return DefWindowProc(hWnd, message, wParam, lParam);
      }
      return 0;
}
```

Лабораторная работа № 18

Тема: Разработка кода программного модуля, создающего окно, в котором выводится синусоида, движущаяся влево, как график функции y = sin (x + t)

Цель: Обучение низкоуровневому программированию и использованию таймера через разработку оконного приложения, выводящего график функции y = sin (x + t)

Время выполнения: 80 минут

Проверяемые результаты обучения

Перечень объектов контроля и оценкиНаименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание 1.

Создать проект Windows-приложения:

- Открыть приложение Visual C++

- Создать проект Windows-приложения:

выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 *Project -> ввести имя проекта в поле Project name "……." -> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK.*

Создание исполняемого модуля Си++-программы для Windows:

- Ввести в поле редактирования файла исходный текст программы, содержащийся в ПРИЛОЖЕНИИ 1 на 6 странице данной лабораторной работы.
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;
- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

.

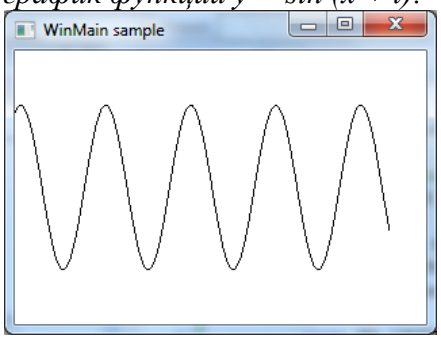
Методические указания

Данная работа создана на основе статьи **Дмитрия Федоркова на сайте:**

- http://procoder.info/index.php/entry/category/statii/ **Автор: Дмитрий Федорков**

Предлагается отредактированный текст статьи для ознакомления:

Напишем простую программу: окно, в нем – синусоида, которая движется влево, как график функции y = sin (x + t):



Если кликнуть мышкой по окну, анимация приостановится, или наоборот продолжится. Чтобы было проще разобраться, сразу приводится весь исходный код, а потом комментируются ключевые места. Попробуйте самостоятельно модифицировать разные части программы, пробуйте оптимизировать программу.

В программе добавляется заголовочный файл **<cmath>***, который нужен для расчета синусоиды, и* **<windows.h>***, который содержит все функции WinAPI. Строчка*

#define WIN32_LEAN_AND_MEAN

отключает некоторые редко используемые функции и ускоряет компиляцию.

Описывается прототип функции **WindowProc()**.

HDC *– контекст устройства рисования. Эта переменная глобальная, потому что используется в обеих функциях. Надо добавить, что буква "H" в типе данных WinAPI (в "HDC") обычно означает "Handle" (хэндл), т.е. переменную, дающую доступ к самым разным устройствам, объектам и прочим сущностям WinAPI. Хэндл – представляет собой обычный указатель, работа с которым зависит от контекста (от типа переменной).*

*Теперь главное (main) – точка входа. В консольных программах функция main может возвращать либо void, либо int, а также может иметь или не иметь аргументы (int argc, char **argv). Итого 4 варианта.*

В нашем случае используется функция WinMain(), которая может иметь только такой вид, как в примере. Слово WINAPI (которое подменяется препроцессором на __stdcall) означает, что аргументы функции передаются через стек. Аргумент HINSTANCE hInstance — хэндл текущего процесса, который бывает нужен в некоторых ситуациях. Назначение следующего аргумента HINSTANCE hPrevInstance весьма смутное, известно только, что эта переменная всегда равна NULL.*

*Аргумент LPSTR lpCmdLine – командная строка. В отличие от консольного main (int argc, char **argv), эта строка не разделена на отдельные аргументы и включает имя самой программы (что-нибудь типа "C:\WINDOWS\system32\format.com C: \u"). Далее int nCmdShow определяет параметры окна, указанные, например, в свойствах ярлыка (это будет нужно при создании окна).*

Перейдем, наконец, к выполняемому коду. В первую очередь нам нужно создать окно. Структура WNDCLASS хранит свойства окна, например текст заголовка и значок. 4-ре из 9-ти полей структуры должны быть нулевые, поэтому сразу инициализируем ее нулями. Далее CS_HREDRAW | CS_VREDRAW означает, что окно будет перерисовываться при изменении размера окна.

wc.hInstance задаёт текущий процесс (тут-то и понадобился этот аргумент из WinMain). Еще также нужно явно указать мышиный курсор, иначе, если это поле оставить нулевым, курсор не будет меняться, скажем, при переходе с границы окна на само окно (попробуйте сами). wc.lpfnWndProc – это адрес функции, которая будет обрабатывать все события. Такие как нажатие клавиши, движение мыши, перетаскивание окна и т. д. После знака "=" просто указываем имя нашей функции. Позже мы напишем эту функцию, которая и будет определять реакцию программы на интересующие нас события.

WNDCLASS *– это не само окно, а класс (форма), экземпляр которого и будет нашим окном. Но перед созданием окна нужно зарегистрировать в системе этот класс. Задаем его имя в системе CMyWnd и регистрируем класс.*

Функция создания окна CreateWindow(). Обратите внимание – все строковые аргументы предваряются буквой L, что означает, что они – юникодовые. Для многих функций WinAPI существует по два варианта: обычный ANSI и юникодовый. Соответственно они имеют суффикс A или W, например CreateWindowA и CreateWindowW. Если вы посмотрите определение функции в <windows.h>, то увидите что-то типа #define CreateWindow CreateWindowW. Вместо CreateWindow() мы можем явно вызывать CreateWindowA() с обычными строками (без приставки L).

Описание функций **GetDC()** *и* **ShowWindow() не требуется.**

Дальше начинается самое интересное – работа с событиями. Для начала создадим таймер, который будет генерировать соответствующее событие 65 раз в секунду (фактически максимальная частота, по крайней мере для Windows XP). Если вместо последнего аргумента SetTimer() написать имя подходящей функции, она будет вызываться вместо генерации события.

Далее идет то, что называется message loop – цикл обработки событий. Мы принимаем событие и обрабатываем его. В нашем случае можно убрать TranslateMessage(&msg), но эта функция понадобится, если на основе этого примера кто-нибудь будет создавать более сложную программу (с обработкой скан-кодов клавиатуры). Если мы получаем событие выхода программы, то GetMessage() возвращает ноль. В случае ошибки возвращается отрицательное значение. В обоих случаях выходим из цикла и возвращаем код выхода программы.

Теперь займемся функцией обработки событий WindowProc(). Эта функция вызывается при любом событии. Какое именно сейчас у нас событие – определяется аргументом message. Дополнительные параметры (например, координаты мыши в событии "мышь двинулась") находятся в аргументах wParam и lParam. В зависимости от того, чему равно message, мы совершаем те или иные (или вообще никакие) действия, а потом в любом случае вызываем DefWindowProc, чтобы не блокировать естественные реакции окна на разные события.

Имена констант message говорят сами за себя.

При событии WM_PAINT рисуем белый прямоугольник, а на нём — чёрную синусоиду.

На каждый WM_TIMER смещаем фазу синусоиды и перерисовываем ее. На клик мыши запускаем или останавливаем движение, при этом, если нажать обе кнопки одновременно, то фаза увеличится ровно на 1, для чего здесь и убран break. При изменении размера окна

мы обновляем переменные Width и Height за счёт того, что в lParam хранятся новые размеры.
Всегда нужно вручную обрабатывать событие WM_DESTROY, иначе окно не будет закрываться. Наша программа закрывается также при нажатии клавиши <Escape>.
Знание принципов работы операционной системы иногда бывает очень полезным, особенно, при оптимизации программы. Не говоря уже о том, что знание WinAPI избавит вас от «изобретения велосипеда».

Задание:

Прочитать статью и ответить на вопросы:

1. Для чего используется данная команда:

#define WIN32_LEAN_AND_MEAN

2. Какие глобальные переменные используются в программе?
3. Для чего используется структура WNDCLASS?
4. Какая функция создаёт таймер?
5. Какая структура используется при обработке событий?
6. Для чего организован цикл обработки событий?
7. Какая функция предназначена для обработки событий?
8. Какие события обрабатываются в данной функции?

Приложение 1

```
#define WIN32_LEAN_AND_MEAN
        #include <windows.h>
        #include <cmath>
        LRESULT CALLBACK WindowProc (HWND, UINT, WPARAM, LPARAM);
        HDC dc;
        int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nCmdShow)
        {
           // Create window
           WNDCLASS wc = {0};
           wc.style = CS_OWNDC | CS_HREDRAW | CS_VREDRAW;
           wc.lpfnWndProc = WindowProc;
           wc.hInstance = hInstance;
           wc.hCursor = LoadCursor (NULL, IDC_ARROW);
           wc.lpszClassName= "CMyWnd";
           RegisterClass (&wc);
           HWND hWnd = CreateWindow ("CMyWnd", "WinMain sample",
WS_OVERLAPPEDWINDOW, CW_USEDEFAULT, 0, 320, 240, NULL, NULL, hInstance,
NULL);
           dc = GetDC (hWnd);
           ShowWindow (hWnd, nCmdShow);
           // Message loop (timer, etc)
           SetTimer (hWnd, 1, USER_TIMER_MINIMUM, NULL);
           MSG msg;
           while (GetMessage(&msg,NULL,0,0) > 0)// while not WM_QUIT (0) nor some error (-1)
           {
              TranslateMessage (&msg);
              DispatchMessage (&msg);
           }
           return msg.wParam;
```

```
        }
        // Message processing function
        LRESULT CALLBACK WindowProc (HWND hWnd, UINT message, WPARAM
wParam, LPARAM lParam)
        {
           static bool Move = true;
           static int Phase=0, Width, Height;
           switch (message)
           {
           case WM_LBUTTONDOWN:
           case WM_RBUTTONDOWN:
              Move = !Move;
              // no break
           case WM_TIMER:
              if (Move)
                 Phase++;
                 // no break
              else
                 break;
           case WM_PAINT:
              Rectangle (dc, -1, -1, Width+1, Height+1);
        MoveToEx (dc, 0, Height * (0.5 + 0.3*sin(0.1*Phase)), NULL);
        for (int i=0; i<Width; i++)
        LineTo (dc, i, Height * (0.5 + 0.3*sin(0.1*(i+Phase))) );
        break;
        case WM_SIZE:
        Width = LOWORD(lParam),
        Height = HIWORD(lParam);
        break;
        case WM_KEYDOWN:
        if (wParam != VK_ESCAPE)
        break;
        // else no break
        case WM_DESTROY:
        PostQuitMessage (0);
        }
        return DefWindowProc (hWnd, message, wParam, lParam);
        }
```

Лабораторная работа № 19

Тема: Разработка кода программного модуля, создающего окно, в котором выводится анимированный осцилограф на WinAPI в C++

Цель: Обучение низкоуровневому программированию и использованию таймера через разработку оконного приложения, выводящего анимированный осцилограф на WinAPI в C++

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного продукта.	Соответствие текста кода программного модуля требованиям разработки программного продукта	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Применение принципов структурного программирования при разработке программного продукта	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание 1.

Создать проект Windows-приложения:

- Открыть приложение Visual C++
- Создать проект Windows-приложения:

выбрать меню File -> выбрать пункт New -> выбрать закладку Projects -> отметить тип создаваемого проекта Win32 *Project -> ввести имя проекта в поле Project name "…….” -> выбрать кнопку OK -> выбрать кнопку Finish -> выбрать кнопку OK.*

Создание исполняемого модуля Си++-программы для Windows:

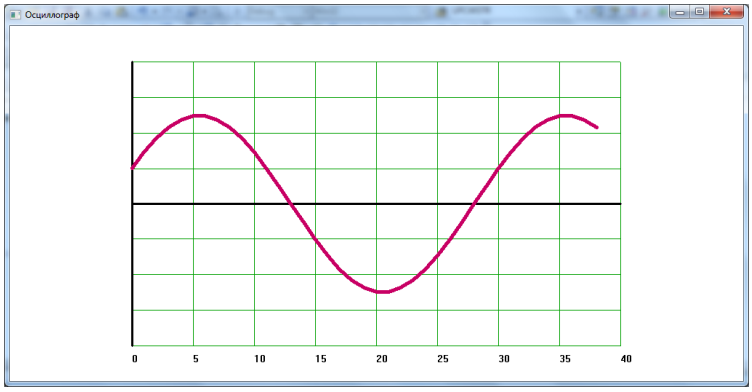
- Сделать установки: *выбрать меню Project -> Properties -> Configuration Properties -> Character set -> выбрать пункт Use Multi-Byte Character Set*;

- Проверить результат работы приложения - запуск исполняемого модуля (последовательность действий: *выбрать меню запуск*).

.

Методические указания

Интерфейс оконного приложения:



Для данной работы использованы материалы статьи сайта http://procoder.info/index.php/entry/category/statii/ **Автор: Олег Кутков**

В этой небольшой статье демонстрируется, как создается окно и как рисовать средствами GDI+. Данный материал будет полезен всем тем, кто хочет разобраться с созданием графический приложений Windows, средствами WinAPI. Анимироваться, в данной статье, будет синусоида, получится своего рода осциллограф.

Для создания этого приложения используется среда Microsoft Visual C++. Запустите IDE и создайте новое Win32 приложение, но укажите опцию, запрещающую генерацию любого кода, нам нужен чистый проект.

Так как мы собираемся использовать WinAPI функции, а так же некоторые математические функции, в начале программы следует подключить два заголовочных файла:

```
#include <windows.h>
#include <math.h>
```

Так же, в программе, нам понадобится значение числа Пи, объявим его здесь же:

```
#define Pi 3.14159265
#define WIN32_LEAN_AND_MEAN // для более быстрой компиляции
```

Каждое Windows приложение имеет так называемую оконную функцию обратного вызова. Это особая функция, которая не вызывается непосредственно в приложении, ее вызывает операционная система, отсюда и название функции. Вызов это функции происходит каждый раз, когда приложению приходит какое-либо сообщение: перерисовать окно, нажата кнопка, приложение закрыто. Данная функция будет реализована чуть ниже, но что бы ее можно было вызывать из любой точки программы, объявим ее в начале как прототип функции:

```
LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);
```

Функции передаются параметры, указывающие какому окну адресуется это сообщение, а также, какое именно сообщение пришло. Объявим еще две вспомогательные переменные:

```
TCHAR szTitle[] = "Осциллограф";
TCHAR szWindowClass[] = "oscill";
```

Это две строки, первая – текст, который будет отображен в заголовке окна, вторая – имя класса окна (это имя выбирается самостоятельно программистом).

Готовим тело приложения…

Вот мы и подобрались к самой главной функции Windows приложения – **WinMain**. Эта функция выполняет роль, аналогичную роли функции main. **WinMain** принимает ряд аргументов:

- **HINSTANCE hInstance** – дескриптор приложения, присваеваемый операционной системой;
- **HINSTANCE hPrevInstance** – параметр, ныне не используемы, оставленный для совместимости с очень старыми приложениями;
- **LPSTR lpCmdLine** – строка, содержащая аргументы запуска приложения, аналог **argv[]**;
- **int nCmdShow** – режим показа главного окна (свернутое, развернутое, по умолчанию).

Общее объявление функции выглядит как:
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) { }
Параметр **WINAPI** перед функцией обозначает, что функция является особой **WINAPI** функцией и нужен операционной системе. Далее, в самой функции, нам следует объявить три переменные, дескритор окна, системное сообщение и структура окна:

```
HWND hWnd;
MSG msg;
WNDCLASSEX wcex;
```

После объявления переменных, следует заполнить поля структуры, ниже показано как, с комментариями:

```
wcex.cbSize = sizeof(WNDCLASSEX);  //размер структуры
wcex.style = CS_HREDRAW | CS_VREDRAW; //задаем стиль окна, подробнее смотрите в MSDN
wcex.lpfnWndProc = (WNDPROC)WindowProcedure;  //указываем оконную процедуру
wcex.cbClsExtra = 0;
wcex.cbWndExtra = 0;
wcex.hInstance = hInstance;  //указываем дескриптор приложениея
wcex.hIcon = LoadIcon(NULL, IDI_APPLICATION);  //устанавливаем иконку приложения по умолчанию
wcex.hCursor = LoadCursor(NULL, IDC_ARROW); //устанавливаем курсор по умолчанию
wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW+1); //задаем цвет окна
wcex.lpszMenuName = 0;     //меню окна - нет меню
wcex.lpszClassName = szWindowClass;  //указываем класс окна
wcex.hIconSm = LoadIcon(NULL, IDI_APPLICATION);  //загружаем иконку окна
```

Далее необходимо зарегистрировать класс окна, с обязательной проверкой результата:

```
if(!RegisterClassEx(&amp;wcex))
{
  MessageBox(hWnd, "Ошибка регистрации класса окна", "Ошибка", IDI_ERROR || MB_OK);
  return 1;
}
```

Теперь пришло время создания окна. Для этого будем использовать функцию **CreateWindow**. Ниже показано, как создать обычное окно, с координатами по умолчанию. Тут так же следует проводить проверку:

```
hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW,
CW_USEDEFAULT, 0, CW_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

if(!hWnd)
{
   MessageBox(hWnd, "Ошибка создания окна", "Ошибка", IDI_ERROR || MB_OK); //в случае
чего - говорим об ошибке
  return 1;
}
```
Теперь окно можно показать на экране:

```
ShowWindow(hWnd, nCmdShow);
```
Мы подобрались к самому концу функции, здесь нас ждет очень важный код – именно тут запускается цикл обработки сообщений операционной системы:

```
while(GetMessage(&amp;msg, NULL, 0, 0))
{
  TranslateMessage(&amp;msg);
  DispatchMessage(&amp;msg);
}
```
Это цикл, с помощью функции **GetMessage**, выбирающий следующее сообщение из очереди сообщений и выполняющий его преобразование и обработку. Цикл заканчивается, как только приходит сообщение **WM_QUIT** и **GetMessage** возвращает **false**.
В самом конце следует написать
return msg.wParam;
Функция **WinMain** завершена.
Теперь на очереди реализация вышеобъявленной функции **WindowProcedure**. В ней происходит обработка всех сообщений и выполнение соответствующих действий. Сразу следует сообщить, так как в нашем приложении будет отрисовываться анимация в окне — в данной функции объявлены необходимые переменные и обработчики сообщений. Ниже представлен скорректированный код (от форумчанина **DomiNick**) всей функции с комментариями:

```
LRESULT CALLBACK WindowProcedure (HWND hWnd, UINT message, WPARAM wParam,
LPARAM lParam)
{
  RECT rect;
  static int offset = 0;
  switch (message)
  {
    case WM_CREATE:
      SetTimer(hWnd, 1, 150, NULL); // "включаем" таймер
      return 0;
    case WM_TIMER:
      GetClientRect(hWnd, &amp;rect);
      InvalidateRect(hWnd, &amp;rect, true);
      UpdateWindow(hWnd);
      ++offset;
      return 0;
    case WM_PAINT:
      PAINTSTRUCT ps;
      HDC hdc;
```

```
      hdc = BeginPaint(hWnd, &amp;amp;ps);
      DrawDiagram(hWnd, hdc, offset);
      EndPaint(hWnd, &amp;amp;ps);
      return 0;
    case WM_DESTROY:
      PostQuitMessage(0);
      return 0;
    default:
      return DefWindowProc(hWnd, message, wParam, lParam);
   }
   return 0;
}
```

Думаю, что все тут наглядно и понятно. Как вы заметили, в сообщении **WM_PAINT** происходит вызов функции **DrawDiagram(hWnd, hdc, offset)** – это нестандартная функция и нам следует ее реализовать. Ей передаются, в качестве параметров, дескриптор окна, дескриптор устройства вывода, а так же новое значение смещения для синусоиды. Для того, чтобы вызывать функцию в этом месте, мы должно объявить ее ранее, что и сделаем, добавьте, в самом верху, после **LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM)** объявление:

```
void DrawDiagram(HWND hwnd, HDC hdc, int offset);
```

И в конце концов самая большая и сложная функция программы – функция рисования синусоиды. Её заголовок уже приведен выше. На самом деле данная функция рисует не только синусоиду, он так же отвечает за отрисовку вертикальных и горизонтальных линий координатной сетки, а так же числовые отметки осей абсцис и ординат. Пусть это будет напряжение и время.

Начать функцию следует с объявление важных констант – координаты рисования сетки, максимальные, минимальные значения, а так же два массива, содержащие текст – числа, которые будут нарисованы возле осей. Так же тут вызываются четыре **API** функции, задающие необходимые параметры рисования, подробнее о них Вы можете прочесть в **MSDN**.

```
RECT rect;
GetClientRect(hwnd, &amp;amp;rect);
const int xVE = rect.right - rect.left;
const int yVE = rect.bottom - rect.top;
const int xWE = xVE;
const int yWE = yVE;
double nPixPerVolt = yVE / 1000.0;
double nPixPerMs = xVE / 60.0;

SetMapMode(hdc, MM_ISOTROPIC);
SetWindowExtEx(hdc, xWE, yWE, NULL);
SetViewportExtEx(hdc, xVE, -yVE, NULL);
SetViewportOrgEx(hdc, 10*nPixPerMs, yVE/2, NULL);

const int tMin = 0;
const int tMax = 40;
const int uMin = -400;
const int uMax = 400;
const int tGridStep = 5;
const int uGridStep = 100;
```

```
int x, y;
int u = uMin;
int xMin = tMin * nPixPerMs;
int xMax = tMax * nPixPerMs;

char* xMark[] = {"0", "5", "10", "15", "20", "25", "30", "35", "40"};
char* yMark[] = {"-40", "-30", "-20", "-10", "0", "10", "20", "30", "40"};
```
Пока оставьте все как есть, потом, изменяя числовые значения, Вы сможете наблюдать за отрисовкой графика и координатной сетки. Далее создадим наше «перо», которым будет осуществляться рисование линй, так же зададим ему цвет.

```
HPEN hPen0 = CreatePen(PS_SOLID, 1, RGB(0, 160, 0));
HPEN hOldPen = (HPEN)SelectObject(hdc, hPen0);
```
Теперь выполним отрисовку сетки – линии ординат и соответсвущющих числовых меток. Для этого сначала переместимся в определенную точку, с помощью функции **MoveToEx(hdc, x, y, NULL)**, а затем нарисуем, «пером», линию в другую точку, с помощью **LineTo(hdc, x, y)**. Рисование текста выполняется с помощью **TextOut(hdc, x, y, string, strlen(string))**:

```
for(int i = 0; i < 9; ++i) {
  y = u * nPixPerVolt;
  MoveToEx(hdc, xMin, y, NULL); //перемещаемся в заданную точку
  LineTo(hdc, xMax, y); //рисуем туда линию
  TextOut(hdc, xMin-40, y+8, yMark[i], strlen(yMark[i])); //выводим текст
  u += uGridStep;
}
```
Теперь выполним небольшие вычисления:

```
int t = tMin;
int yMin = uMin * nPixPerVolt;
int yMax = uMax * nPixPerVolt;
```
И аналогично нарисуем ось абсцис:

```
for(int a = 0; a < 9; ++a) {
  x = t * nPixPerMs;
  MoveToEx(hdc, x, yMin, NULL); //перемещаемся в заданную точку
  LineTo(hdc, x, yMax); //рисуем туда линию
  TextOut(hdc, x, yMin-10, xMark[a], strlen(xMark[a])); //выводим текст
  t += tGridStep;
}
```
Сетка нарисована, теперь нужно нарисовать сами оси, для этого выберем другое «перо»:

```
HPEN hPen1 = CreatePen(PS_SOLID, 3, RGB(0, 0, 0)); //создаем кисть
SelectObject(hdc, hPen1);
```
И с помощью уже знакомых нам функций нарисуем оси:

```
MoveToEx(hdc, 0, 0, NULL); LineTo(hdc, xMax, 0);
MoveToEx(hdc, 0, yMin, NULL); LineTo(hdc, 0, yMax);
```
Теперь самое интересное, отрисовка графика функции. Вновь берем “перо”:

```
HPEN hPen2 = CreatePen(PS_SOLID, 5, RGB(200, 0, 100));
SelectObject(hdc, hPen2);
```

Сначала код с комментариями, затем некоторые пояснения:

```
int tStep = 1; //задаем шаг графика
double radianPerx = 2 * Pi / 30;  вычисляем угол радиан
const double uAmplit = 250; //задаем амплитуду
t = tMin;
MoveToEx(hdc, 0, ((uAmplit * sin(t * radianPerx - offset)) * nPixPerVolt), NULL); //вычисляем начальную точку
while(t <= tMax) {  //до достижения максимального значения x
  u = uAmplit * sin(t * radianPerx - offset); //вычисляем синус и точку, куда рисовать линию
  LineTo(hdc, t * nPixPerMs, u * nPixPerVolt); //рисуем линию
  t += tStep;
}
SelectObject(hdc, hOldPen);
```

Сначала выполняются необходимые вычисления аргументов функции синуса, затем вычисляется собственно синус и в полученную точку осуществляется переход, с помощью MoveToEx. Затем запускается цикл, который, поточечно, начиная с точки, куда мы только что перешли (если-бы этого не сделали, то у синусоиды-бы появилась некрасивая «тянучка» в начале) рисует линию графика. цикл прерывается, как только достигается максимальное значение, заданное выше, в константах.

Задание:

Прочитать статью и ответить на вопросы:

1. Для чего используется данная команда:

```
#define WIN32_LEAN_AND_MEAN
```

2. Какие глобальные переменные используются в программе?
3. Для чего используется структура WNDCLASS?
4. Какая функция создаёт таймер?
5. Какая структура используется при обработке событий?
6. Для чего организован цикл обработки событий?
7. Какая функция предназначена для обработки событий?
8. Какие события обрабатываются в данной функции?

Лабораторная работа № 20

Тема:

Проектирование структуры документа с технической документацией программного продукта низкоуровневого программирования под Windows с использованием библиотеки Программного интерфейса приложений (Application Program Interface, API), 32-разрядного подмножества Win32 API. Этапы проектирования структуры документа**.**

Цель:

Разработать техническую документацию для программного продукта низкоуровневого программирования под Windows с использованием библиотеки Программного интерфейса приложений (Application Program Interface, API), 32-разрядного подмножества Win32 API.

Время выполнения: 120 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата
У 4. Оформлять документацию на программные средства.	Текстовый документ, выполненный согласно общим положениям о стандартах документирования

	программных средств
У 5. Использовать инструментальные средства для автоматизации оформления документации.	Использование текстового редактора.
З 3. Основные принципы отладки и тестирования программных продуктов	Изложение требований к тестированию, определение граничных точек, контроль задания на граничных точках, тестирование программы как чёрного ящика, как белого ящика, пошаговое тестирование, восходящее тестирование, нисходящее тестирование
З 4. Методы и средства разработки технической документации	Наличие титульного листа, постановки задачи, спецификаций, перечня входных и выходных данных, блок-схемы, сценария отладки программы, инструкции пользователя.

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание.
Для готового программного модуля, разработанного как Win32 API-приложение, создать документ, содержащий техническую документацию.

Методические указания:

Документация должна содержать необходимые сведения по установке, обеспечению надёжной работы продукта, справочное пособие для пользователя, демонстрационные версии, примеры документов, создаваемых при помощи данного программного продукта, обучающие программы.
Этапы разработки программной документации:

1. техническое задание (назначение, область применения программы, требования, предъявляемые к программе);
2. текст программы (запись программы с необходимыми комментариями);
3. описание программы (сведения о логической структуре и функционировании программы);
4. пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений);
5. эксплуатационные документы.

1. Техническое задание. Требования к содержанию и оформлению.
ГОСТ 19.201-78 ЕСПД.

Техническое задание (ТЗ) содержит совокупность требований к ПС и может использоваться как критерий проверки и приёмки разработанной программы. Полно составленное и принятое заказчиком и разработчиком ТЗ является одним из основополагающих документов проекта ПС.

В техническое задание включают:

- введение (наименование, краткая характеристика области применения программы);
- основания для разработки (документы, на основании которых ведётся разработка, организация, утвердившая документы, дата утверждения, наименование и обозначение темы разработки);
- назначение разработки (функциональное и эксплуатационное назначение программы);

- требования к программе и программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приёмки.

Наиболее существенной частью технического задания является раздел "требования..." В этом разделе приводятся:

- требования к функциональным характеристикам (состав выполняемых функций, организация входных и выходных данных, временные характеристики);
- требования к надёжности (обеспечение устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа);
- требования к информационной и программной совместимости (требования к информационным структурам на входе и выходе, методам решения, исходным кодам, языкам программирования и программным средствам; требования к защите информации);
- требования к составу и параметрам технических средств;
- требования к программной документации.

Данный раздел может содержать требования к маркировке, упаковке, транспортировке и хранению, а также условия эксплуатации.

2. Текст программы

Кроме явно описанных в техническом задании требований, следует придерживаться общепринятых правил разработки программ с учётом выбранной парадигмы программирования:

1. Программа не должна содержать избыточные элементы (все элементы программы адекватны поставленной задаче: нет циклов, массивов и т. п. элементов, без которых можно обойтись).
2. Алгоритм должен быть структурирован: для функционального стиля программирования - адекватное разбиение на функции (процедуры), для объектно-ориентированного - адекватная иерархия классов. Каждая функция (метод класса) должна реализовывать ровно одно действие.
3. У функций (методов классов) должны быть параметры. Следует избегать использования в функциях глобальных переменных.
4. Программа должна аккуратно использовать память: работать с динамическими массивами, в ней не должно быть неиспользуемых блоков памяти, лишних переменных.
5. Должны проверятся диапазоны вводимых пользователем значений и параметров, передаваемых между модулями программы.
6. При использовании в программе каких-либо готовых компонент (библиотечных функций, классов) если функция или метод класса может завершиться неудачей, необходимо обязательно проверять это, не полагаясь на незначительность вероятности такого события.
7. Программа должна быть конфигурируема (важные параметры программы следует выделить в единый блок).

Текст программы представляет собой символическую запись на исходном или промежуточном языке или символическое представление машинных кодов. Текст программы оформляется моноширинным шрифтом (Courier, Lucida Console и т. п.):

1. Количество операторов на строчке должно = 1.

2. Все операторы, входящие в составной оператор, должны быть сдвинуты вправо на одинаковое количество позиций, при этом операторные скобки (т. е. то, что ограничивает составной оператор), относящиеся к одному блоку, должны располагаться следующим образом: открывающая скобка должна находиться на той же строчке, что и оператор, открывающий блок, а закрывающая должна находиться в той же колонке, с которой начинается оператор, открывающий блок. Допускается располагать открывающую скобку на строке, следующей за оператором, открывающим блок, в той же колонке, с которой начинается этот оператор.

```
If (0) {
    Bbbbbb
    }
```

3. Строка исходного текста программы должна целиком располагаться в одной типографской строке (до 80 символов в зависимости от шрифта). Несоблюдение этого правила говорит о слишком большой вложенности блоков, что означает неудачный алгоритм или структуру программы. В таком случае рекомендуется переосмыслить структуру программы, ввести дополнительные функции, заменив какие-то большие части кода их вызовами, переделать алгоритм и т.п.
4. Если синтаксис языка позволяет, желательно **отделять знаки операций пробелами от операндов**. Как и в обычном тексте, после запятых должен следовать пробел.

```
A > b          (a > b ) && (b < c)
```

5. Определения функций или логические части программы следует отделять друг от друга пустыми строками.
6. **Идентификаторы** (названия переменных, типов, подпрограмм) **должны быть значимыми настолько, чтобы читающий текст программы мог понимать их смысл без присутствия рядом автора**. При необходимости объявление переменной или типа может сопровождаться комментарием.
7. **Текст программы должен содержать комментарии, отражающие функциональное назначение того или иного блока программы, структуру программы**.

3. Описание программы. ГОСТ 19.402-78 ЕСПД.

Документ **"Описание программы"** содержит:

- общие сведения (обозначение и наименование программы, программное обеспечение, необходимое для функционирования программы, языки программирования, на которых написана программа);
- функциональное назначение (классы решаемых задач, сведения о функциональных ограничениях на применение);
- описание логической структуры (алгоритм программы, используемые методы, структура программы с описанием составных частей и связи между ними);
- используемые технические средства (типы ЭВМ и устройств, которые используются при работе программы);
- вызов и загрузка (способ вызова программы с соответствующего носителя данных);
- входные данные (характер, организация и предварительная подготовка входных данных, а также их формат, описание и способ кодирования);
- выходные данные (характер и организация выходных данных, а также их формат, описание и способ кодирования).

Описание логической структуры программы следует сопровождать блок-схемой программы.

Документ "Описание программы" может содержать также схемы данных, схемы взаимодействия программ, схемы ресурсов системы и проч., оформленные в соответствии с ГОСТ 19.701-90.

4. Пояснительная записка. ГОСТ 19.404-78 ЕСПД.
Программный документ "Пояснительная записка" составляется на стадии эскизного или технического проектов программы. Как правило, на стадии рабочего проекта не используется.
Включает разделы:
1.Введение.
2.Назначение и область применения.
3.Технические характеристики.
4.Ожидаемые технико-экономические показатели.
5.Источники, используемые при разработке.

Введение содержит наименование программы и обозначение темы разработки, документы, на основе которых ведётся разработка.
В назначении и области применения указывают назначение программы, краткую характеристику области применения программы.
Технические характеристики содержат:
Постановка задачи на разработку программы, описание применяемых математических методов и различных ограничений, связанных с выбранным математическим аппаратом.
Описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи, возможного взаимодействия программы с другими программами.
Описание и обоснование выбора метода организации входных и выходных данных.
Описание и обоснование выбора состава технических и программных средств на основе проведённых расчётов и анализов, распределение носителей данных, которые использует программа.

5. Эксплуатационные документы

К эксплуатационным документам относят:

- описание применения (сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств);
- руководство системного программиста (сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения);
- руководство программиста (сведения для эксплуатации программы);
- руководство оператора (сведения для обеспечения общения оператора с вычислительной системой в процессе выполнения программы);
- описание языка (описание синтаксиса и семантики языка);
- руководство по техническому обслуживанию (сведения для применения тестовых и диагностических программ при обслуживании технических средств)

Описание применения

Документ "Описание применения" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (возможности, основные характеристики, ограничения области применения);

- условия применения (требования к техническим и программным средствам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера);
- описание задачи (указываются определения задачи и методы её решения);
- входные и выходные данные.

Руководство системного программиста. ГОСТ 19.503-78 ЕСПД.

Документ "Руководство системного программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания системным программистом. Документ состоит из следующих разделов:

- общие сведения о программе (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);
- структура программы (сведения о структуре, взаимосвязи между модулями программы и с другими программами);
- настройка программы (настройка на состав технических средств, выбор функций и т. п.);
- проверка программы (способы и методики проверки, контрольные примеры, методы прогона, результаты);
- дополнительные возможности;
- сообщения системному программисту (тексты сообщений, выдаваемых в ходе выполнения настройки, проверки программы, в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Руководство программиста. ГОСТ 19.504-78 ЕСПД.

Документ "Руководство программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания программистом. Документ состоит из следующих разделов:

- назначение и условия применения программы (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);
- характеристики программы (временные характеристики, режимы работы, средства контроля правильности выполнения и т. п.);
- обращение к программе (способы передачи управления и параметров данных);
- входные и выходные данные (формат и кодирование);
- сообщения (тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Руководство оператора. ГОСТ 19.505-78 ЕСПД.

Документ "Руководство оператора" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (информация, достаточная для понимания функций программы и её эксплуатации);
- условия выполнения программы (минимальный и/или максимальный набор технических и программных средств и т. п.);

- выполнение программы (последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы; описываются функции, форматы и возможные варианты команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды);
- сообщения оператору (тексты сообщений, выдаваемых оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Описание языка. ГОСТ 19.506-78 ЕСПД.

1. Общие сведения
2. Элементы языка
3. Способы структурирования программы
4. Средства обмена данными
5. Встроенные элементы
6. Средства отладки программы

Общие сведения содержат назначение и описание общих характеристик языка, его возможностей, основных областей применения.
Элементы языка - описывается синтаксис и семантика базовых и составных элементов языка.
Способы структурирования программы - описываются способы вызова процедур передачи управления и другие элементы структурирования программы.
Средства обмена данными - описываются языковые средства обмена данными (средства ввода/вывода, средств внутреннего обмена данными)
Встроенные элементы - описываются встроенные в язык элементы: функции, классы и т.д. и правила их использования)
При описании средств отладки необходимо описание имеющихся в языке средств отладки программ, семантики этих средств, дать рекомендации по их применению.

Макет документации к программному продукту:
Содержание:

1. Техническое задание
2. Текст программы
3. Описание программы
4. Пояснительная записка
5. Эксплуатационные документы.

Техническое задание.

ГОСТ 19.201-78 ЕСПД.
Постановка задачи:
Разработать программу, решающюю алгебраическое уравнение вида:
$ax^2 + bx + c = 0$ ($a \neq 0$), где
a–коэффициент при x^2 (первый коэффициент);
b–коэффициент при x (второй коэффициент)
c–свободный член.

Если в≠0, с≠0, то квадратное уравнение полное и корни квадратного уравнения находят по формуле:

$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$, где D = b^2 – 4ac (дискриминант)

Если D<0, то уравнение не имеет корней.
Если D=0, то уравнение имеет равные корни

$x_1 = x_2 = \frac{-в}{2a}$

Программный продукт должен представлять собой Win32 API-приложение 32-разрядного подмножества низкоуровневого программирования под Windows с использованием библиотеки Программного интерфейса приложений (Application Program Interface, API).

Разработать графический интерфейс, позволяющий организовать ввод и вывод данных на уровне элементов управления.
Коэффициенты a, в, с вводятся с помощью элементов управления edit.
Для начала решения квадратного уравнения используется элемент управления button.
Для завершения работы приложения используется элемент управления button.
Входные данные: коэффициенты a, в, с
Выходные данные: корни уравнения x_1, x_2.

Если дискриминант уравнения отрицательный, выдать сообщение: «Уравнение не имеет корней». Если первый коэффициент уравнения = 0, выдать сообщение: «Уравнение не квадратное».

Текст программы

(запись программы с необходимыми комментариями согласно требованиям структурного программирования)

Описание программы

ГОСТ 19.402-78 ЕСПД.

(сведения о логической структуре и функционировании программы)
Указать главную точку входа в программу.
Перечислить используемые глобальные переменные.
Описать используемые элементы управления.
Указать используемые дескрипторы, окна
Директивы препроцессора.
Перечислить обрабатываемые сообщения.
Охарактеризовать входные данные (тип данных, граничные точки).

Пояснительная записка
ГОСТ 19.404-78 ЕСПД.
(схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений)
Блок-схема функции WinMain.
Блок-схема функции WndProc.

Эксплуатационные документы

Описание применения

Руководство системного программиста

ГОСТ 19.503-78 ЕСПД.

Руководство программиста

ГОСТ 19.504-78 ЕСПД.

Руководство оператора

ГОСТ 19.505-78 ЕСПД.

Описание языка
ГОСТ 19.506-78 ЕСПД.

Указать используемые типы переменных, имена используемых функций библиотеки API.
Описать используемые операторы, структуры языка C++.
Указать, с помощью каких операторов организована обработка сообщений

Лабораторная работа № 21
Тема:
Разработка руководства системного программиста Win32 API-приложения.

Цель:
Разработать руководство системного программиста низкоуровневого программирования под Windows с использованием библиотеки Программного интерфейса приложений (Application Program Interface, API), 32-разрядного подмножества Win32 API.

Время выполнения: 120 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата
У 4. Оформлять документацию на программные средства.	Текстовый документ, выполненный согласно общим положениям о стандартах документирования программных средств
У 5. Использовать инструментальные средства для автоматизации оформления	Использование текстового редактора.

документации.	
З 3. Основные принципы отладки и тестирования программных продуктов	Изложение требований к тестированию, определение граничных точек, контроль задания на граничных точках, тестирование программы как чёрного ящика, как белого ящика, пошаговое тестирование, восходящее тестирование, нисходящее тестирование
З 4. Методы и средства разработки технической документации	Наличие титульного листа, постановки задачи, спецификаций, перечня входных и выходных данных, блок-схемы, сценария отладки программы, инструкции пользователя.

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание.

Для готового программного модуля, разработанного как Win32 API-приложение, создать руководство системного программиста.

Методические указания:

Программная документация, включает:

6. техническое задание (назначение, область применения программы, требования, предъявляемые к программе);
7. текст программы (запись программы с необходимыми комментариями);
8. описание программы (сведения о логической структуре и функционировании программы);
9. пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений);
10. эксплуатационные документы

К эксплуатационным документам относят:

- описание применения (сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств);
- руководство системного программиста (сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения);
- руководство программиста (сведения для эксплуатации программы);
- руководство оператора (сведения для обеспечения общения оператора с вычислительной системой в процессе выполнения программы);
- описание языка (описание синтаксиса и семантики языка);
- руководство по техническому обслуживанию (сведения для применения тестовых и диагностических программ при обслуживании технических средств)

Руководство системного программиста. Требования к содержанию и оформлению.

Документ "Руководство системного программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания системным программистом. Документ состоит из следующих разделов:

- общие сведения о программе (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);
- структура программы (сведения о структуре, взаимосвязи между модулями программы и с другими программами);
- настройка программы (настройка на состав технических средств, выбор функций и т. п.);
- проверка программы (способы и методики проверки, контрольные примеры, методы прогона, результаты);
- дополнительные возможности;
- сообщения системному программисту (тексты сообщений, выдаваемых в ходе выполнения настройки, проверки программы, в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Лабораторная работа № 22
Тема:
Разработка пояснительной записки программного продукта Win32 API-приложения.
Цель:
Разработать пояснительную записку как часть технической документации для программного продукта низкоуровневого программирования под Windows с использованием библиотеки Программного интерфейса приложений (Application Program Interface, API), 32-разрядного подмножества Win32 API.

Время выполнения: 120 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата
У 4. Оформлять документацию на программные средства.	Текстовый документ, выполненный согласно общим положениям о стандартах документирования программных средств
У 5. Использовать инструментальные средства для автоматизации оформления документации.	Использование текстового редактора.
З 3. Основные принципы отладки и тестирования программных продуктов	Изложение требований к тестированию, определение граничных точек, контроль задания на граничных точках, тестирование программы как чёрного ящика, как белого ящика, пошаговое тестирование, восходящее тестирование, нисходящее тестирование
З 4. Методы и средства разработки технической документации	Наличие титульного листа, постановки задачи, спецификаций, перечня входных и выходных данных, блок-схемы, сценария отладки программы, инструкции пользователя.

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание.

Для готового программного модуля, разработанного как Win32 API-приложение, создать пояснительную записку. Согласовать терминологию предметной области. Согласовать компьютерную терминологию.

Методические указания:

Документация должна содержать необходимые сведения по установке, обеспечению надёжной работы продукта, справочное пособие для пользователя, демонстрационные версии, примеры документов, создаваемых при помощи данного программного продукта, обучающие программы.

Программная документация, включает:

1. техническое задание (назначение, область применения программы, требования, предъявляемые к программе);
2. текст программы (запись программы с необходимыми комментариями);
3. описание программы (сведения о логической структуре и функционировании программы);
4. пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений);
5. эксплуатационные документы.

4. Пояснительная записка

Программный документ "Пояснительная записка" составляется на стадии эскизного или технического проектов программы. Как правило, на стадии рабочего проекта не используется.

Включает разделы:

1.Введение.

2.Назначение и область применения.

3.Технические характеристики.

4.Ожидаемые технико-экономические показатели.

5.Источники, используемые при разработке.

Введение содержит наименование программы и обозначение темы разработки, документы, на основе которых ведётся разработка.

В назначении и области применения указывают назначение программы, краткую характеристику области применения программы.

Технические характеристики содержат:

Постановка задачи на разработку программы, описание применяемых математических методов и различных ограничений, связанных с выбранным математическим аппаратом.

Описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи, возможного взаимодействия программы с другими программами.

Описание и обоснование выбора метода организации входных и выходных данных.

Описание и обоснование выбора состава технических и программных средств на основе проведённых расчётов и анализов, распределение носителей данных, которые использует программа.

Лабораторная работа № 23

Тема:

Методы и средства разработки технической документации программного продукта Win32 API-приложения. Руководство пользователя.

Цель:

Разработать руководство пользователя как часть технической документации для программного продукта низкоуровневого программирования под Windows с использованием

библиотеки Программного интерфейса приложений (Application Program Interface, API), 32-разрядного подмножества Win32 API.

Время выполнения: 120 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата
У 4. Оформлять документацию на программные средства.	Текстовый документ, выполненный согласно общим положениям о стандартах документирования программных средств
У 5. Использовать инструментальные средства для автоматизации оформления документации.	Использование текстового редактора.
З 3. Основные принципы отладки и тестирования программных продуктов	Изложение требований к тестированию, определение граничных точек, контроль задания на граничных точках, тестирование программы как чёрного ящика, как белого ящика, пошаговое тестирование, восходящее тестирование, нисходящее тестирование
З 4. Методы и средства разработки технической документации	Наличие титульного листа, постановки задачи, спецификаций, перечня входных и выходных данных, блок-схемы, сценария отладки программы, инструкции пользователя.

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание.
Для готового программного модуля, разработанного как Win32 API-приложение, создать руководство пользователя программного продукта.

Методические указания:
Документация должна содержать необходимые сведения по установке, обеспечению надёжной работы продукта, справочное пособие для пользователя, демонстрационные версии, примеры документов, создаваемых при помощи данного программного продукта, обучающие программы.
Программная документация, включает:

11. техническое задание (назначение, область применения программы, требования, предъявляемые к программе);
12. текст программы (запись программы с необходимыми комментариями);
13. описание программы (сведения о логической структуре и функционировании программы);
14. пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений);
15. эксплуатационные документы.

5. Эксплуатационные документы

К эксплуатационным документам относят:

- описание применения (сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств);
- руководство системного программиста (сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения);
- руководство программиста (сведения для эксплуатации программы);
- руководство оператора (сведения для обеспечения общения оператора с вычислительной системой в процессе выполнения программы);
- описание языка (описание синтаксиса и семантики языка);
- руководство по техническому обслуживанию (сведения для применения тестовых и диагностических программ при обслуживании технических средств)

Основная часть программной документации составляется на стадии рабочего проекта. Необходимость того или иного документа определяется на этапе составления технического задания. Допускается объединять отдельные виды документов.
Эксплуатационный документ "Описание языка" включается в программную документацию, если разработанный программный продукт реализует некий язык программирования, управления заданиями, организации вычислительного процесса и т. п.
Эксплуатационный документ "Руководство по техническому обслуживанию" включается в программную документацию, если разработанный программный продукт требует использования тестовых или диагностических программ.

Описание применения

Документ "Описание применения" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (возможности, основные характеристики, ограничения области применения);
- условия применения (требования к техническим и программным средствам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера);
- описание задачи (указываются определения задачи и методы её решения);
- входные и выходные данные.

Руководство программиста

Документ "Руководство программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания программистом. Документ состоит из следующих разделов:

- назначение и условия применения программы (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);
- характеристики программы (временные характеристики, режимы работы, средства контроля правильности выполнения и т. п.);
- обращение к программе (способы передачи управления и параметров данных);
- входные и выходные данные (формат и кодирование);

- сообщения (тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Руководство оператора

Документ "Руководство оператора" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (информация, достаточная для понимания функций программы и её эксплуатации);
- условия выполнения программы (минимальный и/или максимальный набор технических и программных средств и т. п.);
- выполнение программы (последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы; описываются функции, форматы и возможные варианты команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды);
- сообщения оператору (тексты сообщений, выдаваемых оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

3.1.2 Самостоятельные работы

***Самостоятельная работа № 1 к теме «*Интерфейс Windows приложений.**

Осуществление разработки кода программного модуля на языке С++»
Тема: Внутреннее представление данных в компьютере.
Цель: контроль знания операторов языка С++ на уровне внутреннего представления данных в компьютере.
Задание сохранить в виде файла в общей папке для студентов под своей фамилией.

Таблица ASCII

Таблица ASCII представляет собой кодировку для представления десятичных цифр, латинского и национального алфавитов, знаков препинания и управляющих символов.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Задание 1
Внутреннее представление данных символьного типа:

Методические указания:

Дана строка:
char string[] = "Hello!";
Данная строка представляет собой оператор присваивания переменной символьного типа (массив) значения Hello!. Длина строки = количеству символов 6+1 (признак конца массива \0) . Один символ занимает 1 байт. Под строку "Hello!" необходимо 7 байт или 56 битов.

Элемент массива	Символ	таблица ASCII	Двоичное представление
string[0]	H	72	01001000
string[1]	e	102	01100110
string[2]	l	108	01101100
string[3]	l	108	01101100
string[4]	o	111	01101111
string[5]	!	33	00100001
\0	NULL	000	00000000

Внутреннее представление строки в памяти компьютера:
01001000 01100110 01101100 01101100 01101111 00100001 00000000

Ход работы:
Дана строка:
char a[] = "To say";
1. Что означает данная строка языка программирования C++?
2. Сколько памяти будет отведено под переменную a?
3. Записать внутреннее представление строки в памяти компьютера аналогично описанному выше примеру

++

Задание 2

Внутреннее представление данных целого типа:

1.Чтобы получить внутреннее представление положительного целого числа, небходимо:
Перевести число в двоичную систему счисления;
Полученный результат дополнить слева незначащими нулями до к-разрядов (k – количество бит, отводимых под данный тип числа) .

2.Чтобы получить внутреннее представление отрицательного целого числа, небходимо:
Перевести число в двоичную систему счисления;
Полученный результат дополнить слева незначащими нулями до к-разрядов.
Получить обратный код этого числа заменой 0 на 1, а 1 на 0.
К полученному числу прибавить 1.

```
1. int integer1, integer2, integer3, sum;
2. integer1 = 15;
3. integer2 = 16;
4. integer3 = 17;
5. sum = integer1 + integer2 + integer3;
```

Получить внутреннее представление данных целого типа (переменные sum, integer1,integer2, integer3) при выполнении каждого оператора описанного выше кода.

Под переменную целого типа (int) отводится 2 байта (16 битов, k = 16).
В результате выполнения 1 оператора будет выделено 4 ячейки по 16 битов для переменных integer1, integer2, integer3, sum.
В результате выполнения второго оператора в ячейку, отведённую под переменную integer1, будет введено значение 15 в двоичном коде:

0000000000001111
В результате выполнения третьего оператора в ячейку, отведённую под переменную integer2, будет введено значение 16 в двоичном коде:
0000000000010000
В результате выполнения четвёртого оператора в ячейку, отведённую под переменную integer3, будет введено значение 17 в двоичном коде:
0000000000010001
В результате выполнения пятого оператора в ячейку, отведённую под переменную sum, будет введено значение 48 в двоичном коде:

```
  0000000000001111
+0000000000010000
_________________
  0000000000011111

  0000000000011111
+ 0000000000010001
_________________
  0000000000110000
```

Задание 3

```
int i, j;
j =10 ;
for (i = 0; i<=3; i++) {j = j - 1} ;
```

1. Написать блок-схему к данному коду программы языка С++
2. Показать, как меняется содержимое ячеек, отведённых под переменные i и j? (внутреннее представление переменных i и j в памяти компьютера при выполнения каждого оператора).

Задание 4

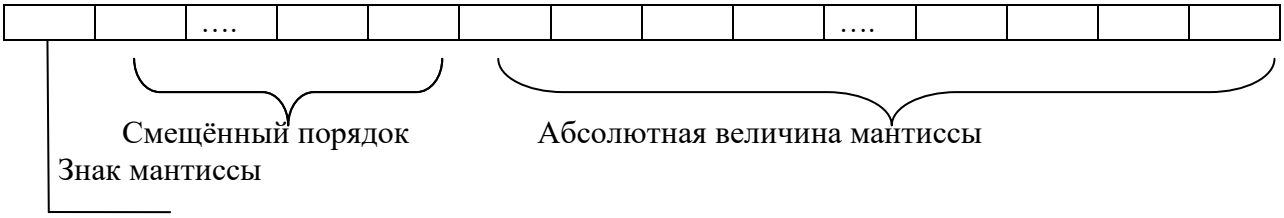
```
 int counter, grade, total;
 total = 0;
 counter = 0;
 grade = 3;
 while (grade != 0)
 {
        total = total + grade;
        counter = counter + 1;
        grade = grade - 1;
 }
```

Показать, как меняется содержимое ячеек, отведённых под переменные counter, grade, total при выполнении каждого оператора.

Задание 5

Внутреннее представление данных вещественного типа (число с плавающей запятой):

Стандартная форма представления вещественного числа:
32-разрядное нормализованное число со знаком, 8 разрядным смещенным порядком и 24 разрядной мантиссой (старший бит мантиссы, всегда равный 1, не хранится в памяти и размер поля, выделенного для хранения мантиссы, составляет только 23 разряда)



float average;
average = 1.125;

Представить содержимое ячейки, отведённой под переменную average в двоичном виде (выделить порядок и мантиссу)

Задание 6

Написать оператор языка программирования C++, в результате которого описывается переменная символьного типа с именем average.

Задание 7

Написать оператор языка программирования C++, в результате которого описывается переменная целого типа с именем counter и эта переменная приобретает значение 12.

1.Сколько памяти будет отведено под переменную counter?

2.Представить значение переменной в том виде, в котором оно будет храниться в ячейке, отведённой под переменную counter.

***Самостоятельная работа № 2 к теме «*Интерфейс Windows приложений.**

Осуществление разработки кода программного модуля на языке C++»

Тема: Низкоуровневое программирование под Windows с использованием библиотеки Программного интерфейса приложений (Application Program Interface, API), 32-разрядного подмножества Win32 API.

Цель: изучение структуры приложения Win32 API и операторов языка программирования C++, используемых в приложении Win32 API.

Задача: разобрать структуру приложения Win32 API с детальным изучением основных операторов и структур языка программирования C++.

Методический материал:

Стиль программирования Windows-приложений принципиально отличается от того, который сложился в операционных системах раннего поколения. В MS-DOS программа монопольно владеет всеми ресурсами системы и сама является инициатором взаимодействия с операционной системой.

Операционная система Windows является многозадачной и именно она является инициатором обращения к программе (приложению).

Все ресурсы Windows являются разделяемыми и приложение не может владеть ими монопольно. Приложение должно ждать посылки сообщения операционной системы Windows и лишь после его получения выполнить определённые действия. Затем вновь перейти в режим ожидания очередного сообщения.

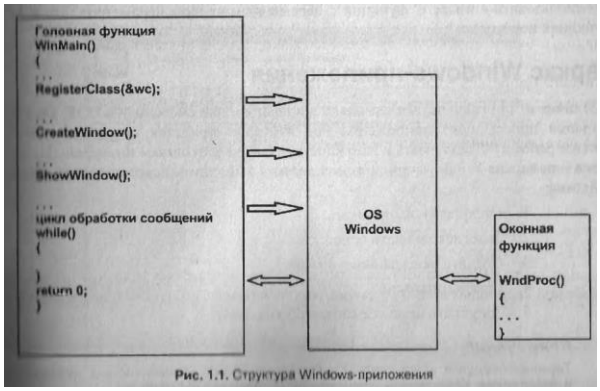


Рис. 1.1. Структура Windows-приложения

Windows генерирует множество различных сообщений (щелчок мыши, нажатие клавиши на клавиатуре…), которые направляются приложению. Задачей программиста является обработка лишь тех сообщений, которые необходимы приложению. Сообщения, которые не обрабатываются приложением, обрабатываются операционной системой Windows.

Разработчиками операционной системы Windows была создана библиотека функций, при помощи которых происходит взаимодействие приложения с операционной системой (функции Программного интерфейса приложений Application Program Interface- **API**).

Множество этих функций предназначено для графического вывода на дисплей и принтер. Их называют – интерфейс графических устройств (Graphics Device Interface, **GDI**).

Библиотека API-функций разрабатывалась в расчёте на то, что эти функции можно использовать для любого языка программирования, а поскольку разные языки программирования имеют разные типы данных, то были созданы собственные Windows-типы данных, которые приводятся к типам данных других языков программирования. Например, в Windows нет логического типа bool, но есть Windows-тип BOOL, который эквивалентен типу int. В Windows введены синонимы CALLBACK, APIENTRY, WINAPI.

Таблица 2.1. *Наиболее часто используемые типы данных Windows*

Тип	Описание
CALLBACK	Замещает спецификацию FAR PASCAL в функциях обратного вызова
HINSTANCE	32-разрядное беззнаковое целое число, используемое в качестве дескриптора
HDC	Дескриптор контекста устройства
HWND	Дескриптор окна
LPARAM	Используется для описания младшего аргумента сообщения
LPCSTR	32-разрядный указатель на константную строку
LPSTR	32-разрядный указатель на строку
LRESULT	Используется для возвращения значений из оконных процедур
UINT	32-разрядное беззнаковое целое число
WINAPI	Замещает спецификацию FAR PASCAL в описаниях функций API
WPARAM	Используется для описания старшего аргумента сообщения

Таблица 2.2. *Наиболее часто используемые структуры Windows*

Структура	Что определяет
MSG	Параметры сообщения
PAINTSTRUCT	Область окна, требующая перерисовки
WNDCLASS	Класс окна

Каркас Windows-приложения, имеющего возможность работать с оконным интерфейсом:

1.Определить класс окна.

WNDCLASS **wc**; // описание структуры **wc**

2.Зарегистрировать окно

RegisterClass();

3.Создать окно данного класса

CreateWindow();

4.Отобразить окно.

ShowWindow();

UpdateWindow();

5.Запустить цикл обработки сообщений.

```
while(GetMessage(&msg, NULL, 0, 0))
{
        TranslateMessage(&msg);
        DispatchMessage(&msg);
}
```

Вопросы:

1. Операционная система Windows является однозадачной или многозадачной?
2. Приведите пример однозадачной операционной системы.
3. Владеет ли Windows- приложение ресурсами монопольно?
4. Что значит в Windows – разделяемые ресурсы?
5. Когда Windows- приложение может выполнить определённые действия?
6. Что значит - режим ожидания очередного сообщения от операционной системы?
7. Как называется библиотека функций, при помощи которых происходит взаимодействие приложения с операционной системой Windows?
8. Какие действия должно выполнить Windows- приложение, чтобы иметь возможность работать с оконным интерфейсом?

16.2. Описание структур

Рассмотрим следующее описание структуры:

```
struct Card {
   char *face;
   char *suit;
};
```

Ключевое слово **struct** открывает описание структуры **Card**. Идентификатор **Card** является *именем структуры* и используется в C++ для объявления переменных *типа структуры* (в языке C именем типа приведенной выше структуры было бы **struct Card**). Данные (иногда и функции, как в классах), объявленные в фигурных скобках описания структуры, являются *элементами* структуры. Элементы одной структуры должны иметь уникальные имена, но две разные структуры могут включать элементы с одинаковыми именами, не конфликтующими друг с другом. Каждое объявление структуры должно завершаться точкой с запятой.

Структуры могут содержать переменные разных типов данных в отличие от массивов, которые включают только элементы одного типа. Этот факт и

единственное реальное различие между структурами и классами в языке C++ состоит в том, что к элементам структуры по умолчанию доступ является открытым, а к элементам класса — закрытым. Обычно структуры используются для определения записей данных, которые должны быть сохранены в файлах (см. главу 14, «Обработка файлов и ввод-вывод потоков строк»).

Приведенное объявление структуры не предусматривает выделения какой-либо области памяти; более уместно сказать, что объявление структуры создает новый тип данных, который может использоваться в дальнейшем при объявлении переменных типа структуры. Переменные типа структуры объявляются аналогично переменным других типов. Объявление

```
Card a, deck[52[, *c;
```

объявляет **a** — переменную типа структуры **Card**, **deck** — массив с 52 элементами типа **Card** и **c** — указатель на структуру типа **Card**. Переменные заданного типа структуры могут также объявляться с помощью списка разделенных запятыми имен переменных, расположенного между закрывающейся фигурной скобкой и точкой с запятой, завершающей объявление структуры. Например, предыдущее объявление можно включить в описание структуры **Card** следующим образом :

```
struct Card {
   char *face;
   char *suit;
} a, deck[52], *c;
```

Имя структуры является необязательным. Если объявление структуры не содержит ее имени, то переменные структурного типа могут быть объявлены только в описании структуры, а не с помощью их отдельного объявления.

Структура wc, определяющая класс окна приложения Win32 API.

```
WNDCLASS wc;
        // Заполнение структуры WNDCLASS для регистрации класса окна.
        memset(&wc, 0, sizeof(wc));
        wc.lpszClassName=ClassName;
        wc.lpfnWndProc=(WNDPROC)WndProc;
        wc.style=CS_HREDRAW|CS_VREDRAW;
        wc.hInstance=hInstance;
        wc.hIcon=LoadIcon(NULL,IDI_APPLICATION);
        wc.hCursor=LoadCursor(NULL,IDC_ARROW);
        wc.hbrBackground=(HBRUSH)(COLOR_BTNFACE+1);
        wc.lpszMenuName=NULL;
        wc.cbClsExtra=0;
        wc.cbWndExtra=0;

        // Регистрация класса окна.
        RegisterClass(&wc);
```

Задание 1:
Описать тип и элементы структуры wc.

Возвращаемое значение функции

```
int WINAPI WinMain()
{
        WNDCLASS wc;            // Структура для информации о классе окна
        HWND hWnd;        // Дескриптор главного окна приложения
        MSG msg;          // Структура для хранения сообщения
```

…….тело функции WinMain, являющейся главной точкой входа в приложение Win32

return **msg.wParam**;
}

// Создаем главное окно приложения.
Обращение к функции:
hWnd= CreateWindow();

Описание функции CreateWindow находится в библиотеке Программного интерфейса приложений (Application Program Interface, API):

HWND CreateWindow(…)

{
…….тело функции CreateWindow, создающей окно приложения Win32 API
return …….; // возвращаемым значением функции CreateWindow является сформированный дескриптор окна

}

// Отображаем окно.
ShowWindow(hWnd, nCmdShow);

// Обновляем содержимое клиентской области окна.
UpdateWindow(hWnd);

Задание 2
Перечислить все функции приложения Win32 API, имеющие возвращаемое значение. (см. Приложение 1 на стр. 10)

Цикл обработки сообщений

Цикл обработки начинается с извлечения сообщений из очереди при помощи функции:

BOOL GetMessage(MSG FAR* lpmsg, HWND hwnd, UINT uMsgFilterMin,UINT uMsgFilterMax);

Если при вызове этой функции указать вторым аргументом NULL, то программа будет получать сообщения от всех окон, созданных программой. При помощи параметров uMsgFilterMin и uMsgFilterMax можно отфильтровать сообщения, получаемые программой. Если на их месте передать 0, то программа будет получать все сообщения.

После вызова функции GetMessage приложение получает структуру msg типа MSG с информацией о сообщении. Структура MSG содержит информацию об окне, которому предназначено сообщение, численное значение самого сообщения, время и координаты (для сообщений от мыши) возникновения сообщения, а также два дополнительных параметра wParam и lParam, смысл и значение которых зависят от особенностей сообщения.

Каждое получаемое приложением сообщение (за исключением WM_QUIT) направлено одному из окон приложения. Поскольку приложение не должно прямо вызывать функцию обработки окна, для передачи сообщения нужному окну используется функция:

LONG DispatchMessage(const MSG FAR* lpmsg);

Эта функция передает msg обратно в Windows. Windows отправляет сообщение для его обработки соответствующей оконной процедуре – таким образом Windows вызывает оконную процедуру.

После того, как оконная функция обработает сообщение, Windows возвращает управление в приложение к следующему за вызовом DispatchMessage коду, цикл обработки сообщений в очередной раз возобновляет работу, вызывая GetMessage. Если поле message сообщения msg, извлеченного из очереди сообщений, равно любому значению, кроме WM_QUIT, то функция GetMessage возвращает ненулевое значение. Сообщение WM_QUIT, извлеченное из очереди приложения заставляет прервать цикл обработки сообщений.

Перед вызовом функции DispatchMessage могут быть помещены специальные функции, производящие над помещенными в очередь сообщениями какие-то действия. Например, преобразование некоторых сообщений, полученных с помощью клавиатуры, в более понятные сообщения можно при помощи функции

BOOL TranslateMessage(const MSG FAR* lpmsg);

Задание 3
Ответить на вопросы::
Для чего создаётся цикл обработки сообщений?
Какая функция создаёт структуру msg типа MSG?
Какая информация содержится в структуре msg?

Структура множественного выбора switch – case.

Структура множественного выбора switch состоит из ряда меток case и необязательной метки default (умолчание).

Хороший стиль программирования 2.24

Хотя предложения **case** и **default** могут размещаться в структуре **switch** в произвольном порядке, стоит учесть практику качественного программирования — помещать **default** в конце.

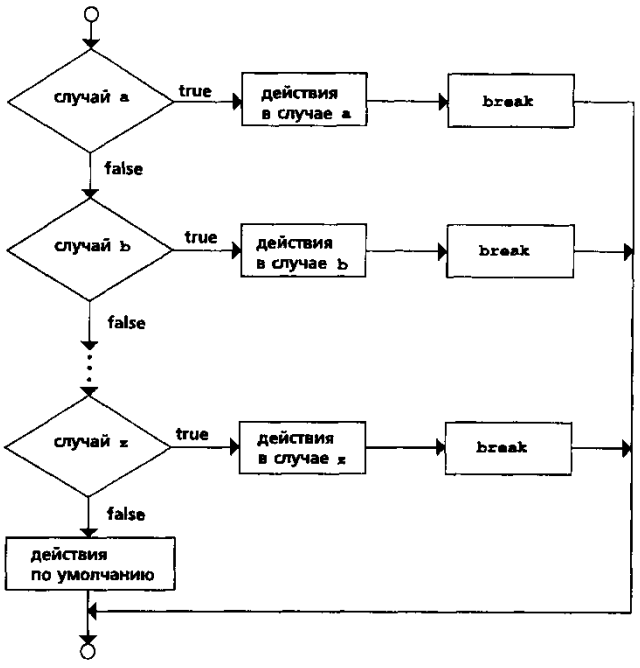


Рис. 2.23. Структура множественного выбора **switch**

Операторы break и continue изменяют поток управления. Когда оператор break выполняется в структурах while, for, do/while или switch, происходит немедленный выход из структуры. Программа продолжает выполнение с первого оператора после структуры. Обычное назначение оператора break — досрочно прерывать цикл или пропустить оставшуюся часть структуры switch

```
switch(msg)
{
        case WM_PAINT:
                {
                        HDC hDC;
                        PAINTSTRUCT ps;
                        hDC=BeginPaint(hWnd,&ps);
                        TextOut(hDC,20,20,str,strlen(str));
                        EndPaint(hWnd,&ps);
                 }; break;

                // Нажата левая клавиша мыши в клиентской области окна.
              case WM_LBUTTONDOWN:
                {
                        MessageBox(hWnd, "32-bit aplication", "Window"," ",
                                MB_OK|MB_ICONINFORMATION);

                }; break;

               case WM_DESTROY:
                {
                        PostQuitMessage(0);
                }; break;

                // Необработанные сообщения передаем в стандартную
                // функцию обработки сообщений по умолчанию.
              default: return DefWindowProc(hWnd, msg, wParam, lParam);
}
```

Задание 4
Описать, с помощью каких операторов и каким образом в Win32 API организована обработка сообщения?
Как добавить обработку нового сообщения?

Обработка сообщений

Операционная система способна генерировать сообщения с номерами до 0x400, номера же от 0x400 и далее зарезервированы для пользовательских сообщений. В файле включений `winuser.h` размещены макроимена системных сообщений. Этот файл вызывается неявно через файл `windows.h`. Рассмотрим технику обработки наиболее распространенных сообщений Windows.

Нажатие клавиши

При нажатии любой алфавитно-цифровой клавиши на клавиатуре вырабатывается сообщение WM_CHAR.

Примечание

На самом деле генерируются сообщения о нажатии и отпускании клавиши WM_KEYDOWN, WM_KEYUP и лишь затем WM_CHAR.

Чтобы обработать это сообщение, необходимо добавить в переключатель оконной функции еще одну строку альтернативы:

```
case WM_CHAR:
```

и описать необходимые действия для обработки нажатия клавиши.

Сообщение мыши

Нажатие на кнопки мыши приводит к ряду сообщений, вот некоторые из них:

- WM_LBUTTONDOWN — нажатие на левую кнопку мыши;
- WM_LBUTTONUP — отпускание левой кнопки мыши;
- WM_RBUTTONDOWN — нажатие на правую кнопку мыши;
- WM_RBUTTONUP — отпускание правой кнопки мыши;
- WM_MOUSEMOVE — перемещение мыши.

Создание окна

При создании окна генерируется сообщение WM_CREATE еще до его отображения что позволяет производить некоторые начальные установки. Мы продемонстрируем использование этого сообщения в следующем примере.

Задание 5
Какие ещё сообщения обрабатываются в Win32 API

Метод получения дескриптора контекста рабочей области окна

Этот метод используется при обработке сообщения WM_PAINT. Применяются две функции: BeginPaint и EndPaint.

Для этих двух функций требуется дескриптор окна (передаваемый в оконную процедуру как параметр) и адрес переменной типа структуры PAINTSTRUCT, определяемой в оконной процедуре.

Вызов BeginPaint заполняет поля этой структуры, а также возвращает дескриптор контекста рабочей области окна, который должен запоминаться в переменной типа HDC. Вызов функции EndPaint освобождает дескриптор контекста рабочей области окна.

При обработке в оконной процедуре сообщения WM_PAINT вызовы функций BeginPaint и EndPaint должны обязательно вызываться парой.

Задание 6
Опишите, какие действия выполняются компьютером в результате работы каждого оператора данного кода:

```
HWND hWnd;
char *str ="Good buy!";
hDC hDC;
PAINTSTRUCT ps;
hDC=BeginPaint(hWnd,&ps);
  TextOut(hDC,20,20,str,strlen(str));
EndPaint(hWnd,&ps);
```

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Применение функций в приложени Win32 API	**1 балл**
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования.	Использофвание унукций в приложени Win32 API для применения принципов структурного программирования	**1 балл**
ПО 2. Разработки кода программного продукта на основе готовой спецификации на уровне модуля.	Применение функций в приложени Win32 API	**1 балл**

За верное выполнение работы выставляется – 3 балла.

За не полностью выполненную работу выставляется – 1 балл.

За невыполненную работу выставляется – 0 баллов.

3.1.3. Тренинговые задания для усвоения профессиональных терминов

Тема: «Контроль знаний функций WIN32 API»

Наименование функции	**Назначение**
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)	Функция WinMain – главная точка входа в приложение **WIN32 API**
hWndMain=FindWindow(ClassNameMW, NULL)	позволяет найти окно верхнего уровня по имени класса или по заголовку окна
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam);	Описание функции главного окна
GetStockObject(WHITE_BRUSH);	Кисть для окон
RegisterClass(&wc);	Регистрация класса окна.
hWndMain = CreateWindow("MainWindow","OVERLAPPEDWINDOW",WS_OVERLAPPEDWINDOW,	создание главного перекрывающегося окна

CW_USEDEFAULT,CW_USEDEFAULT, CW_USEDEFAULT,CW_USEDEFAULT, NULL,NULL,hInstance,NULL);	Координаты углов окна
ShowWindow(hWndMain, nCmdShow);	Сделать окно видимым
UpdateWindow(hWndMain);	Обновить окно
GetMessage(&msg, NULL, 0, 0))	Функция получает сообщение из очереди, выдает false при выборке из очереди сообщения WM_QUIT
TranslateMessage(&msg);	Преобразование некоторых сообщений, полученных с помощью клавиатуры
DispatchMessage(&msg);	Отправляем сообщение оконной процедуре
hDC=BeginPaint(hWndMain,&ps);	Получить контекст окна
TextOut(hDC,20,20,str,strlen(str));	// Нарисовать текст
EndPaint(hWndMain,&ps);	Освободить контекст окна
MessageBox(hWndMain, "32-bit aplication","Window",MB_OK|MB_ICONINFORMATION);	Выводит сообщение в виде отдельного окна
PostQuitMessage(0);	следует в очередь сообщений пиложения послать сообщение WM_QUIT
DefWindowProc(hWndMain, msg, wParam, lParam);	Необработанные сообщения передаем в стандартную функцию обработки сообщений по умолчанию
hbrush=CreateHatchBrush(HS_DIAGCROSS,RGB(255,0,0)); hbrush2=CreateSolidBrush(RGB(0,0,255));	создадим кисти для окрашивания областей
x=LOWORD(lParam); y=HIWORD(lParam);	определим текущую координату курсора(передается в lParam)
PtInRegion(l,x,y)	если курсор в левом кружке, то устанавливаем флаг TRUE

Rectangle(hdc,x1,y1,x2,y2); //
l=CreateEllipticRgn(x1-8,y1+8,x1+8,y1-8);
r=CreateEllipticRgn(x2-8,y2+8,
x2+8,y2-8);
FillRgn(hdc,l,hbrush);
FillRgn(hdc,r,hbrush);
FrameRgn(hdc,r,hbrush2,1,1);FrameRgn(hdc,l,hbrush2,1,1);
	рисуем прямоугольник
	создаем круглые области в углах
	для наглядности закрашиваем их и рисуем им границы

InvalidateRect(hwnd,NULL,TRUE); UpdateWindow(hwnd);	требуем перерисовки окна
SetScrollRange(hWndMain,SB_VERT,min_sb, max_sb,TRUE);	Создание полосы прокрутки
SetScrollPos(hWndMain,SB_VERT,pos_sb,TRUE);	Создание полосы прокрутки
HDC hDC=GetDC(hWnd); RECT r; GetClientRect(hWnd,&r); TEXTMETRIC tm; GetTextMetrics(hDC,&tm);	Получить контекст окна получение размеров рабочей области окна получение информации о текущем шрифте
SetTimer(hWnd,TIMER_SEC,1000,NULL) SetTimer(hWnd,TIMER_MIN,6000,NULL)	Включить таймер
KillTimer(hWnd,TIMER_SEC);	Выключить таймер
GetWindowText(hWndEdit_a,str_a,6);	Получить из окна редактора, имеющего дескриптор hWndEdit_a, введённое значение в текстовую переменную str_a.
float a = atof(str_a); float d = a * a;	Преобразовать число в виде текста из текстовой переменной str_a в вещественную переменную a
sprintf(str5, "%.3g", d);	Преобразовать число из вещественной переменной d в текстовую переменную str5.

SetWindowText(hWndEdit_d,str5);	Разместить значение текстовой переменной str5 в редакторе, имеющем дескриптор hWndEdit_d.

Тема: Организация работы с приложением в многозадачной операционной системе

Выражение	Действие
В однозадачной операционной системе	программа монопольно владеет всеми ресурсами системы и является инициатором взаимодействия с операционной системой
В многозадачной операционной системе	все ресурсы являются разделяемыми, операционная система является инициатором обращения к программе.
Разделяемые ресурсы - это	экран монитора, принтер, сканер, память, модем …
Сообщение-это	структура данных, сформированная операционной системой для данного приложения в ответ на действие - щелчок кнопки мыши, нажатие клавиши на клавиатуре.
Приложение находится в режиме «ожидания сообщения»	если работающее приложение ожидает посылки сообщения операционной системы, выполняет действие и опять ожидает сообщения
Сообщения, которые не обрабатывает приложение	обрабатывает операционная система стандартным образом
Сообщения, которые обрабатывает приложение	приводят к выполнению действия согласно постановке задачи.

Практическое задание
Тема: Низкоуровневое программирование под Windows

Термин	Определение
Программирование в Win32 API	Низкоуровневое программирование под Windows с использованием библиотеки Программного интерфейса приложений API, 32-разрядного подмножества Win32 API.
API	Application Program Interface – библиотека функций программного интерфейса приложений
Каркас Windows-приложения, создающего окно	1. Определить класс окна; 2. Зарегистрировать окно; 3. Создать окно данного класса; 4. Отобразить окно; 5. Запустить цикл обработки сообщений OS Windows
Цикл обработки сообщений	извлечение сообщений из очереди при помощи функции GetMessage() в цикле (while() { })
Точка входа в программу	функция **WinMain**, вызываемая операционной системой при запуске приложения в Windows
Дескриптор экземпляра приложения	уникальное число, идентифицирующее программу, когда она работает под Windows
Дескриптор окна	уникальное число, идентифицирующее окно данного приложения
Контекст устройства	структура данных, связанная с конкретным устройством ввода / вывода информации (принтер или дисплей).
Элементы управления	“button” - кнопка, ”edit” - редактор,

	“static” - статическое дочернее окно управления, “listbox” - для создания одноколоночных и многоколоночных списков, имеющих вертикальные и горизонтальные полосы просмотра, “combobox” - комбинация списка и однострочного редактора, “scrollbar ” - создание полос прокрутки.

3.1.4 Внеаудиторные самостоятельные работы

Внеаудиторная самостоятельная работа № 1
Текст задания

Подготовка реферата по одной из тем:
1.Новые концепции программирования.
2.Объектно-ориентированное программирование.
3.Каркас Windows приложений.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов.

Внеаудиторная самостоятельная работа № 2
Текст задания

Написать программу, строящую прямоугольник в центре окна.

Написать программу движения шарика в окне.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов.

Внеаудиторная самостоятельная работа № 3
Текст задания

Написать программу, которая строит окно как шахматную доску.

Написать программу движения шарика в окне.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов.

Внеаудиторная самостоятельная работа № 4

Текст задания

Создать текстовую программу вывода строки текста

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов.

Внеаудиторная самостоятельная работа № 5

Текст задания

Создать программу, фильтрующую выбор имени файла.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов.

Внеаудиторная самостоятельная работа № 6

Текст задания

Создать программу просмотра текстового файла с пунктом меню для чтения текстового файла в кодировке DOS и Windows..

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов.

Внеаудиторная самостоятельная работа № 7
Текст задания
Создать программу, изменяющую активность пунктов меню.
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов.

Внеаудиторная самостоятельная работа № 8
Текст задания
Создать программу просмотра текстового файла с пунктом меню для чтения текстового файла в кодировке DOS и Windows.
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов.

Внеаудиторная самостоятельная работа № 9
Текст задания
Написать программу, изображающую шахматную доску, где каждая клетка будет дочерним окном.
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки	Выполнение разработки приложения поэтапно	2 балла

программного обеспечения; программирования;		
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов.

Внеаудиторная самостоятельная работа № 10

Текст задания

Создать программу построения графика, предусматривающую блокировку пункта меню до тех пор, пока не будет загружен файл с данными.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 11

Текст задания

Создать программу построения графика, предусматривающую блокировку пункта меню до тех пор, пока не будет загружен файл с данными. В программе построения графика предусмотреть вывод координат во всплывающем окне.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 12

Текст задания

Написать текстовую программу для работы со списком при помощи окна EditBox.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии	Разработка приложения согласно требованиям	

структурного и объектно-ориентированного	структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 13
Текст задания

Написать программу, которая делит окно на четыре равные части и выводит в каждой части растровые изображения.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 14
Текст задания

Создать программу «просмотрщик» графических файлов.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 15
Текст задания

Написать программу с использованием наложения изображений.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 16

Текст задания

Написать программу, которая в стандартном диалоговом окне выбирает шрифт и выводит по центру окна текст.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.

За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 17

Текст задания

Создать DLL – библиотеку для работы с С-строкой, включив в неё аналоги стандартных функций.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.

За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 18

Текст задания

Построить демонстрационную задачу для использования библиотеки DLL при явном связывании.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.

За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 19

Текст задания

Построить демонстрационную задачу для использования библиотеки DLL при неявном связывании.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 20

Текст задания

Создать DLL-библиотеку с галереей рисунков одинакового размера.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 21

Текст задания

Подготовка реферата по теме: Основные правила оформления программной документации.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У4 Оформлять документацию на программные средства	Документация к приложению выполнена согласно предъявляемым требованиям	2 балла
З4 Методы и средства разработки технической документации.	При разработке документации использованы средства разработки документации	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 22

Текст задания

Подготовка реферата по теме: Оформление текстового и графического материала.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)

У4 Оформлять документацию на программные средства	Документация к приложению выполнена согласно предъявляемым требованиям	2 балла
З4 Методы и средства разработки технической документации.	При разработке документации использованы средства разработки документации	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 23
Текст задания
Подготовка реферата по теме: «Виды программных документов».
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У4. Оформлять документацию на программные средства	Документация к приложению выполнена согласно предъявляемым требованиям	2 балла
З4 Методы и средства разработки технической документации.	При разработке документации использованы средства разработки документации	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

3.1.5 Задания для получения допуска к сдаче дифференцированного зачёта по МДК .01.01

Задание1.

Постановка задачи:

Разработать оконное приложение для операционной системы Windows с использованием низкоуровневых средств программирования - набора базовых интерфейсов программирования приложений для OS Windows Win32 API (Application programming interface), которое позволяет входные данные вводить из файла input.txt, а в выходной файл output.txt выводить значение true, если указанное высказывание является истинным, и false в противном случае:

1) Сумма двух первых цифр заданного четырёхзначного числа равна сумме двух его последних цифр;
2) Сумма цифр данного трёхзначного числа N является чётным числом;
3) Точка с координатами (x,y) принадлежит части плоскости, лежащей между прямыми x = m, x = n (m<n);
4) Квадрат заданного трёхзначного числа равен кубу суммы цифр этого числа;
5) Целое число N является чётным двузначным числом;
6) Треугольник со сторонами a, b, c является равносторонним;
7) Треугольник со сторонами a, b, c является равнобедренным;
8) Среди чисел a, b, c есть хотя бы одна пара взаимно противоположных чисел;
9) Числа c и b выражают длины катетов одного прямоугольного треугольника, а c и d – другого. Эти треугольники являются подобными;
10) Даны три стороны одного и три стороны другого треугольника. Эти треугольники равновеликие, т.е. имеют равные площади;
11) Данная тройка натуральных чисел a, b, c являются тройкой Пифагора,т.е. имеют равные площади;
12) Все цифры данного четырёхзначного числа N различны;
13) Данные числа x, y являются координатами точки, лежащей в первой координатной четверти;
14) (x1, y1) и (x2, y2) – координаты левой верхней и правой нижней вершин прямоугольника; точка A (x,y) лежит внутри этого прямоугольника или на одной его сторон;
15) Число c является средним арифметическим чисел a и b;
16) Цифры данного четырёхзначного числа N образуют строго возрастающую последовательность;
17) Цифры данного трёхзначного числа N являются членами арифметической прогрессии;
18) Цифры данного трёхзначного числа N являются членами геометрической прогрессии;
19) Данные числа c и d являются соответственно квадратом и кубом числа a;

20) Цифра M входит в десятичную запись четырёхзначного числа N;

21) Данное четырёхзначное число читается одинаково слева направо и справа налево;

22) Сумма двух натуральных чисел кратна 2;

23) Произведение натуральных чисел a и b кратно числу c;

24) Сумма двух действительных чисел a и b является целым числом, т.е. дробная часть суммы равна нулю;

25) Данное натуральное число a кратно числу b, но не кратно числу c.

Задание 2. Разработать документацию к оконному приложению, используя методические указания, описанные в лабораторной работе № 20.

Критерии оценки программного продукта согласно показателям качества:

1. Показатели надежности программного продукта:
   − устойчивость функционирования;
   − работоспособность.
2. Показатели сопровождения:
   − структурность.
3. Простота конструкции.
4. Наглядность.
5. Повторяемость.
6. Показатели удобства применения.
7. Легкость освоения.

Критерии оценки:

«5»	«4»	«3»	«2»
Созданный программный продукт разработан в полном соответствии с показателями качества.	Созданный программный продукт имеет несоответствие по одному из показателей: простота конструкции.	Созданный программный продукт имеет несоответствие по двум показателям: простота конструкции, показатели удобства применения.	Созданный программный продукт не соответствует более 2 показателям.

3.I. 6 Задания для промежуточной аттестации (МДК.01.01)

3.1.6.1 Вопросы для освоения теоретического курса

1. Из каких двух функций состоит простейшее Windows- приложение?
2. Что такое сообщение?
3. Формат сообщения, параметры сообщения.
4. Какие есть функции для отправки сообщений? Чем они отличаются?
5. Источники сообщений.
6. Какая информация передается в сообщении WM_COMMAND?
7. Из каких двух функций состоит минимальный цикл обработки сообщений?
8. Как создать окно?
9. Какие типы окон существуют?
10. Чем тип окна отличается от класса окна?
11. Как дочерние окна взаимодействуют с родительским окном?

12. Как удалить окно с экрана?
13. Как удалить окно из памяти?
14. Какие органы управления Вы знаете?
15. Что такое идентификатор органа управления?
16. Где идентификатор органа управления связывается с самим элементом управления?
17. В каком сообщении используется идентификатор органа управления?
18. Как ОС Windows информирует приложение о изменении содержимого его окон?
19. Понятие контекста устройства, как его получить?
20. Как место получения контекста влияет на характер отображения информации в окне?
21. Какая функция является точкой входа в Windows-программу?
22. Что такое дескриптор копии приложения? Что он идентифицирует?
23. Какие действия обычно выполняет функция WinMain?
24. Для чего необходимо регистрировать класс окна?
25. Какие характеристики класса окон можно задать при его регистрации? Какие поля структуры, описывающей класс окна, являются наиболее важными?
26. Что такое оконная процедура? Для чего она предназначена? Кто ее вызывает в процессе работы приложения?
27. Может ли приложение создавать окна, принадлежащие классам, которые приложение самостоятельно не регистрировало?
28. Что такое дескриптор окна? Для чего он необходим?
29. После создания окна оно обязательно должно появиться на экране? Что необходимо сделать для отображения окна?
30. Для чего служит цикл обработки сообщений? Откуда приложение извлекает очередное сообщение? Каким образом и кем сообщения создаются?
31. Сколько очередей сообщений существует? Для кого предназначены сообщения?
32. Каким образом цикл обработки сообщений прерывается, давая возможность приложению завершить свою работу?
33. Кто обычно посылает сообщение (и какое), которое прерывает цикл обработки сообщений?
34. Что должно сделать приложение, если оно не обрабатывает некоторое сообщение?
35. Из каких источников оконная процедура получает сообщения?
36. Какое сообщение приходит окну, если часть его рабочей области (или вся она) требует перерисовки? Что такое недействительная область?
37. В каких ситуациях рабочая область может становиться недействительной?
38. Вызовом какой функции должна начинаться обработка сообщения WM_PAINT? Вызовом какой функции она должна заканчиваться? Для чего вызываются эти функции?
39. Как обычно реагирует стандартное главное окно приложения на получение сообщения о его удалении? Что при этом происходит с приложением?
40. Когда приходит сообщение об удалении окна? Что произойдет, если остальные окна приложения будут реагировать на это сообщение так же, как и главное окно?
41. Что определяет (характеризует) стиль класса окна? Что определяет стиль окна?
42. Как задается стиль класса окна?
43. Для чего задаются стили класса CS_HREDRAW, CS_VREDRAW?
44. Для чего используется стиль класса CS_DBLCLKS?
45. Какая характеристика используется для уточнения внешнего вида и поведения окна, создаваемого на базе какого-либо класса?
46. Какой внешний вид и поведение обычно характерен для перекрывающихся окон (назвать стиль класса)?
47. Что такое окно-владелец и подчиненное окно? Чем поведение подчиненного окна отличается от поведения обычного перекрывающегося окна?
48. Для чего чаще всего используются временные (всплывающие, popup) окна?

49. Какой внешний вид обычно имеют временные окна (назвать стиль класса)?
50. Где располагается начало системы координат для перекрывающихся и временных окон?
51. Для чего обычно используются дочерние окна?
52. Где располагается начало системы координат для дочерних окон?
53. Какой внешний вид обычно имеют дочерние окна (назвать стиль класса)?
54. Каковы особенности поведения дочерних окон?
55. Каким простым способом можно сообщить Windows о том, что окно должно иметь горизонтальную и/или вертикальную полосу прокрутки? В чем недостатки такого метода?
56. Какое значение имеют понятия диапазон полосы прокрутки и ее текущее положение? Как можно изменять эти характеристики?
57. Какова сфера ответственности Windows в организации работы пользователя с полосой просмотра?
58. Какова сфера ответственности приложения в организации работы пользователя с полосой просмотра?
59. Какие сообщения приходят окну от полос просмотра?
60. Организован ли на системном уровне интерфейс клавиатуры для работы с полосами прокрутки?
61. При помощи какой функции можно определить системные метрики Windows?
62. Когда окно получает сообщение WM_SIZE и какие дополнительные параметры передаются окну вместе с этим сообщением?
63. Для чего предназначена функция GetClientRect?
64. Когда окно получает сообщение WM_MOVE и какие дополнительные параметры передаются окну вместе с этим сообщением?
65. Для чего предназначена функция GetWindowRect?
66. Как определить метрики шрифта, установленного в контексте отображения окна?
67. О чем уведомляется окно, получившее сообщение WM_PAINT?
68. При возникновении каких причин окно получает сообщение WM_PAINT?
69. В каких случаях Windows самостоятельно восстанавливает изображение в рабочей области?
70. Какое сообщение обязательно посылается окну перед отправкой ему сообщения WM_PAINT? Как реагирует на это сообщение оконная функция обработки сообщений по умолчанию?
71. Что произойдет после первого же прихода сообщения WM_PAINT, если функция окна нарисует что-либо в окне во время обработки сообщений, отличных от WM_PAINT?
72. Что такое недействительная область (или областью обновления)?
73. Что происходит при появлении недействительной области окна?
74. Может ли в очереди сообщений находится несколько сообщений WM_PAINT?
75. Как приложение может удалить из очереди сообщение WM_PAINT?
76. При помощи чего приложения могут рисовать на устройствах вывода?
77. Когда приложение может использовать функции BeginPaint и EndPaint?
78. Обязательно или нет передавать необрабатываемые сообщения WM_PAINT в функцию обработки сообщений по умолчанию? Что при этом происходит?
79. Как получить контекст устройства, если необходимо рисовать в рабочей области при обработке отличных от WM_PAINT сообщений?
80. Каким образом приложение может явно потребовать перерисовку всего окна или его части?
81. Что делает функция InvalidateRect?
82. Что является итогом выполнения функции UpdateWindow?
83. Какая функция является “противоположной” для функции InvalidateRect?

84. Что определяет контекст устройства?
85. В каких ситуациях приложение может не создавать контекст устройства вывода?
86. Что такое общий контекст? Как приложение может его получить?
87. Когда для общего контекста следует выполнять настройку атрибутов?
88. Для чего служит контекст класса окна?
89. Следует ли освобождать контекст класса окна после его получения?
90. Когда желательно использовать контекст класса окна?
91. Можно ли для контекста класса окна выполнять настройку большинства атрибутов всего один раз?
92. Какие атрибуты контекста класса окна обязательно следует настраивать после его получения?
93. Когда используется личный контекст? В преимущества и недостатки его использования?
94. Кем используется и что позволяет делать родительский контекст?
95. Какой контекст позволяет осуществлять вывод в нерабочую область окна? Как его получить?
96. Как можно получить информацию об устройстве вывода? Контекст какого типа следует для этого использовать?
97. Какой контекст применятся для представления растрового изображения в памяти? С кем должен быть совместим этот контекст? Как создается такой контекст?
98. Что такое метафайл? Что он позволяет делать? Как получить его контекст?
99. При помощи какого контекста осуществляется вывод изображений на такое устройство, как, например, принтер?
100. При помощи какого контекста приложение может рисовать на всей поверхности экрана?
101. Какие объекты GDI приложение может создавать и использовать в процессе своей работы?
102. Какова последовательность действий, выполняемых приложением, использующим объекты GDI?
103. Что необходимо сделать для того, чтобы функции отрисовки могли использовать тот или иной объект GDI?
104. Что такое предопределенные системой объекты GDI? Как приложение может получить к ним доступ?
105. Для чего применяются перья? Каким образом их можно создавать и использовать?
106. Что такое режим смешивания переднего плана, на что он влияет?
107. Что такое режим фона и цвет фона, что они определяют?
108. Для чего используются кисти? Какие кисти может создать приложение?
109. Что такое начало отсчета кисти? Для чего используется этот атрибут?
110. Как приложение может создать свой шрифт, что оно для этого должно указать функции создания шрифта?
111. Каковы основные моменты работы с растровыми изображениями?
112. Для чего используется технология отсечения?
113. Какую форму может иметь область отсечения?
114. Что происходит, когда пользователь нажимает или отпускает клавишу на клавиатуре?
115. Что такое фокус ввода? Как он связан с понятием активного окна?
116. Может ли приложение, обрабатывающее сообщения WM_SETFOCUS и WM_KILLFOCUS повлиять на приобретение или потерю фокуса ввода окном?
117. Когда в окно поступают аппаратные сообщения клавиатуры, какие типы аппаратных сообщений существуют?

118. Какие сообщения Windows являются аппаратными клавиатурными сообщениями?
119. В чем специфика обработки системных аппаратных клавиатурных сообщений?
120. Что такое виртуальный код клавиши? Что он идентифицирует? Зависит ли этот код от аппаратной реализации клавиатуры?
121. Как приложение может получить информацию о нажатии на определенную клавишу на момент, когда последнее сообщение от клавиатуры было выбрано из очереди?
122. Как приложение может получить информацию о нажатии на определенную клавишу в текущий момент времени?
123. Почему чаще всего приложение обрабатывает не аппаратные сообщения, а символьные? Что такое символьные сообщения, как они формируются?
124. Как следует модифицировать цикл обработки сообщений для того, чтобы приложение могло получать символьные сообщения?
125. Какое сообщение является символьным клавиатурным сообщением и какую дополнительную информацию оно несет с собой?
126. Какие наборы символов использует Windows, чем они отличаются и где применяются?
127. Как можно определить наличие мыши и ее характеристики?
128. Чем является курсор мыши, что такое вершина курсора мыши?
129. Где приложение обычно определяет форму и вид, которые приобретает курсор, когда он находится над поверхностью окна?
130. Какие сообщения могут поступать приложению при работе пользователя с мышью?
131. От чего зависит количество сообщений WM_MOUSEMOVE, которые получает окно приложения?
132. Обязательно ли приложение, получившее сообщение WM_LBUTTONDOWN, получит и сообщений WM_LBUTTONUP? В каких случаях это будет обязательно, а в каких нет?
133. Что необходимо сделать, чтобы окно приложения получало сообщения о двойных щелчках мыши? Что будет происходить в противном случае?
134. Когда приложение получает сообщения мыши, связанные с нерабочей областью окна?
135. В чем заключается процесс захвата мыши окном? Что при этом происходит?
136. Как приложение может установить таймер? Что происходит после его установки?
137. Каким способом можно заставить Windows посылать сообщения таймера обычной оконной процедуре приложения?
138. Каким способом можно заставить Windows посылать сообщения неоконным функциям приложения? Какая это должна быть функция? Почему?
139. Что такое идентификатор таймера? Для чего при обработке сообщений от таймера следует проверять передаваемый вместе с сообщением идентификатор таймера?
140. Являются ли элементы управления окнами? Как они создаются, могут ли они общаться посредством сообщений со своим родительским окном?
141. На базе каких классов создаются стандартные элементы управления Windows?
142. Для чего при создании элемента управления приложение должно сообщить Windows дескриптор родительского окна и идентификатор создаваемого элемента управления?
143. Что такое дескриптор элемента управления и идентификатор элемента управления?
144. Какое сообщение обычно приходит родительскому окну, если что-то происходит с элементом управления? Какую дополнительную информацию несет с собой это сообщение?
145. Что такое код уведомления? Что он идентифицирует?
146. Какими способами родительское окно может передавать сообщения элементам управления? Чем они отличаются?

147. В чем заключается проблема элементов управления и передачи между ними фокуса ввода? В окнах какого класса эта проблема решена на системном уровне?
148. Какие способы применяются для согласования цвета рабочей области и цветов элементов управления?
149. Чем нажимаемая кнопка со стилем BS_DEFPUSHBUTTON функционально отличается от других нажимаемых кнопок?
150. Что такое флажки-переключатели и радио-переключатели? Чем они отличаются и когда обычно применяются?
151. Для чего обычно приложение использует статические элементы управления? Получает ли родительское окно такого элемента от него сообщения?
152. Поля редактирования каких стилей может создать приложение?
153. Какие списки можно создать на базе предопределенного класса “listbox”?
154. На базе какого класса создается выпадающий список? Комбинацией каких других элементов управления он является?
155. Как приложение может создать сколько угодно полос просмотра в своем окне? Какой предопределенный класс при этом используется?
156. Что такое файл ресурсов? Какими способами его можно создавать?
157. На каком этапе создания приложения ресурсы записываются в загрузочный модуль приложения?
158. Какие данные приложение может хранить в виде ресурсов?
159. Какова последовательность работы приложения с таблицей текстовых строк, содержащейся в ресурсах приложения?
160. Как в ресурсы приложения включить изображение пиктограммы?
161. Какие существуют способы загрузки и использования пиктограммы, содержащейся в ресурсах приложения?
162. Какой оператор используется для включения в ресурсы приложения изображения курсора мыши?
163. Какие существуют способы загрузки и использования изображения курсора мыши, содержащегося в ресурсах приложения?
164. Как включить изображение типа bitmap в ресурсы приложения?
165. Как загрузить и использовать изображение типа bitmap?
166. Какие существуют типы меню?
167. Какими способами можно создать меню?
168. Что такое шаблон меню и где он хранится? Можно ли при помощи шаблона меню создавать многоуровневые меню?
169. Как в шаблоне меню определяются такие элементы меню как пункты и подменю?
170. Для чего необходимы идентификаторы пунктов меню?
171. Как указать, что все окна класса по умолчанию должны иметь некоторое меню?
172. Каким способом окно может установить для себя меню, отличное от меню класса окон?
173. Можно ли (и как) меню создавать динамически, в процессе работы приложения?
174. Какие сообщения окну приходят от меню? Чем сообщение WM_COMMAND отличается от сообщения WM_SYSCOMMAND?
175. Чем сообщение WM_COMMAND, приходящее от пункта меню, отличается от этого же сообщения, но пришедшего от элемента управления?
176. Можно ли модифицировать системное меню окна? Как это сделать?
177. В чем состоит особенность обработки сообщений WM_SYSCOMMAND?
178. Как создать использовать независимое плавающее меню? Какие сообщения приходят при выборе его пунктов?
179. Что такое акселераторы? Где они определяются?
180. Как загрузить и использовать таблицу акселераторов? Какие изменения следует внести в цикл обработки сообщений?

181. Какое сообщение приходит окну при нажатии быстрых клавиш? Чем эти сообщения отличаются от аналогичных сообщений от пунктов меню и элементов управления?
182. Является ли диалоговая панель окном Windows? Чем они отличаются от перекрывающихся окон?
183. Какие существуют способы расположения элементов управления на диалоговой панели?
184. Какие шаги следует предпринять для создания диалоговой панели?
185. Какие типы диалоговых панелей существуют? Чем они отличаются друг от друга?
186. Чем отличаются модальные диалоги от системных модальных?
187. Что определяет ресурс шаблона диалога?
188. В каких единицах измерения задаются координаты и размеры элементов управления диалоговой панели?
189. Чему равняются основные единицы диалогового окна?
190. Что такое диалоговая процедура? Является ли она оконной функцией окна диалога?
191. Чем отличается диалоговая процедура от оконной и чем они похожи?
192. Получает ли диалоговая процедура сообщения WM_PAINT, WM_CREATE, WM_DESTROY? Как происходит обработка этих сообщений?
193. Когда в диалоговую процедуру поступает сообщение WM_COMMAND с идентификатором IDCANCEL?
194. Когда в диалоговую процедуру поступает сообщение WM_COMMAND с идентификатором IDOK?
195. Вызовом какой функции создается модальная диалоговая панель?
196. Возвращает ли функция создания модального диалога управление сразу же после вывода диалоговой панели?
197. Что должна сделать диалоговая процедура для того, чтобы модальная диалоговая панель прекратила свою работу?
198. Проходят ли сообщения для модальных диалоговых окон через очередь сообщений приложения?
199. Что следует сделать для всех дочерних окон элементов управления, если к ним необходимо обеспечить доступ с помощью клавиши <Tab>?
200. Что необходимо сделать для обеспечения передачи фокуса от одного переключателя к другому внутри группы?
201. Чем отличается поведение немодального и модального диалоговых окон?
202. Вызовом какой функции создается немодальная диалоговая панель?
203. Возвращает ли функция создания немодального диалога управление сразу же после вывода диалоговой панели?
204. Что необходимо сделать для того, чтобы немодальная диалоговая панель прекратила свою работу?
205. Можно ли прекратить работу немодального диалога функцией EndDialog?
206. Проходят ли сообщения для немодальных диалоговых окон через очередь сообщений приложения?
207. Какие изменения следует внести в цикл обработки сообщений для того, что немодальное диалоговое окно могло получать сообщения, предназначенные для него?
208. Чем являются окна сообщений? При помощи какой функции они создаются?
209. Как можно повлиять на состав элементов управления окна сообщений?
210. Для чего приложение используют стандартные диалоговые панели выбора файла? Какие панели такого типа существуют? Какие функции предназначены для их создания?
211. Для чего предназначен диалог выбора цвета? Как его создать?
212. Что приложение может получить в результате работы диалоговой панели выбора шрифта? Как отобразить эту панель на экране?

213. Какие диалоговые окна предназначены для выбора принтера, его инициализации, для выбора источника и типа бумаги?
214. Какие диалоговые окна предоставляют собой интерфейс, посредством которого пользователь может ввести текстовую строку для поиска и, если необходимо, строку для замены?
215. Какие стандартные диалоги являются модальными, а какие нет? К чему приводит это различие?
216. Как происходит процесс управления памятью (какие функции при этом используются) с помощью библиотечных функций С?
217. Какие функции ядра Windows предназначены для простого управления памятью? Какова последовательность их применения?
218. Что такое перемещаемая и удаляемая память?
219. Какие функции ядра Windows рекомендуется использовать для работы с файлами?
220. Что такое файлы, проецируемые в память? Как их можно использовать?
221. Как работают приложения в многозадачной операционной среде, реализующей невытесняющую многозадачность?
222. Что такое процесс и поток?
223. Как реализована вытесняющая многозадачность?
224. Как приложение может создать еще один поток, какие действия оно должен сделать?
225. Какие средства синхронизации потоков может применять приложение?
226. Как один поток может уведомить о чем-то другой поток, умеющий обрабатывать сообщения?
227. Если вторичный поток не имеет возможности обрабатывать события, то при помощи чего другой поток может ему сообщить о чем-либо?

3. 1.6.2. Практические задания для МДК.01.01

1. Написать программу **в WINAPI** , строящую прямоугольник в центре окна. При нажатии на левую кнопку мыши его размеры уменьшаются, а при нажатии на правую кнопку – увеличиваются на 10 пикселов.
2. Решить предыдущую задачу, только размеры должны изменяться автоматически через 1 секунду. Нажатие на левую кнопку мыши меняет направление изменения размеров. Правая кнопка завершает работу.
3. Написать программу движения шарика в окне с отражением от стенок по законам геометрической оптики **в WINAPI**. Начало движения происходит из точки, в которой нажимается левая кнопка мыши. Скорость постоянна. Начальный угол определяется случайным образом. Размер шарика и скорость движения выбрать произвольно.
4. Написать программу **в WINAPI**, которая разрисует окно, как шахматную доску и при нажатии левой кнопкой мыши выведет окно сообщений с именем клетки, где находится курсор в шахматной нотации.
5. Написать программу **в WINAPI**, которая с периодичностью в 0,1 секунды заполняет окно прямоугольниками случайного размера (не превосходящего ¼ окна) случайным цветом.
6. Построить стрелочный секундомер во всё окно. Запускается секундомер левой кнопкой мыши, нажатие любой клавиши останавливает отсчёт, правая кнопка – сброс.
7. Создать тестовую программу вывода строки текста **в WINAPI**, меняя размер шрифта от минимально читаемого размера до 1 дюйма.
8. Написать программу **в WINAPI**, котороя выводит все характеристики шрифта, возвращаемые в структуре TEXTMETRIC. Проверить работу со шрифтом «Times New Roman».
9. Дополнить фильтр выбора имени файла возможностью выбора файлов с расширением dat и без расширения имени.
10. Обработать сообщение о перемещении движка полосы скроллинга SB_THUMBTRACK и колёсика мыши.
11. Написать программу, изображающую шахматную доску, где каждая клетка будет дочерним окном, которое меняет цвет при щелчке левой кнопкой мыши на его поверхности, а при нажатии на правую кнопку появляется всплывающее окно с координатами клетки в шахматной нотации. Всплывающее окно можно создать на элементе управления STATIC.
12. Написать тестовую программу для работы со списком при помощи «приятельского» окна Edit Box.
13. Написать программу для построения круговой диаграммы в отдельном всплывающем окне. Реализовать диалоговое окно для настройки цветов заполнения.
14. С помощью немодального диалогового окна реализовать функцию поиска слова для программы просмотра текстовых файлов. Если слово найдено. Обеспечить прокрутку текста до необходимой позиции и выделить найденное слово цветом
15. Написать программу, которая делит окно на 4 равные части и выводит в каждой четверти растровое изображение, растягивая его на весь выделенный прямоугольник. При изменении размеров окна размеры изображений должны корректироваться.
16. Создать программу «просмотрщика» графических файлов. Использовать диалоговое окно выбора имени bmp-файлов. Организовать скроллинг, если изображение не помещается в окне.
17. В стандартном диалоговом окне выбрать шрифт и вывести по центру окна произвольный тест, повторяя базовой линией письма контур синусоиды. Использовать поворот координат для наклона каждого выводимого символа.
18. Вывести в расширенный метафайл растровое изображение и сравнить размер emf – файла с исходным bmp-файлом.

19. Создайте DLL-библиотеку UserString для работы с С-строкой, включив в неё аналоги стандартных функций: strlen(), strcpy(), strrev()…
20. Постройте демонмтрационную задачу для использования созданной библиотеки при явном и неявном связывании.
21. Создайте библиотеку, содержащую набор ресурсов: иконку, курсор, растровое изображение. Постройте демонстрационную задачу, использующую ресурсы этой DLL-библиотеки.
22. Создайте DLL-библиотеку с галереей рисунков одинакового размера. В число экспортируемых переменных включите название галереи и количество изображений. Окно создаваемого приложения заполните этими рисунками.
23. При помощи утилиты dumpbin просмотрите разделы экспорта и импорта созданных библиотек.
24. Создать простейшее приложение WinAPI, которое выводит второе окно при нажатии правой кнопки мыши в клиентской области окна функции.
25. Создать простейшее приложение WinAPI, которое демонстрирует основные стили окон (окно верхнего уровня, всплывающее окно с родителем, всплывающее окно без родителя, дочернее окно).
26. Создать простейшее приложение WinAPI, использующее окна различных стилей (главное, всплывающее и дочернее). Зарегистрировать отдельные классы окон ("MainWindows", "PopupWindows" и "ChildWindows"), предусмотрев для каждого класса собственный цвет фона и собственную функцию окна.
27. Создать простейшее приложение WinAPI, использующее окон различных стилей (главное, всплывающее) зарегистрировать отдельные классы окон ("MainWindows", "PopupWindows "), предусмотрев для каждого класса собственный цвет фона и собственную функцию окна. Добавить в окно приложения вертикальную полосу прокрутки. Включить стиль WS_VSCROLL в описание стиля создаваемого функцией CreateWindow окна.
28. Создать простейшее приложение WinAPI, использующее окон различных стилей (главное, всплывающее) зарегистрировать отдельные классы окон ("MainWindows", "PopupWindows "), предусмотрев для каждого класса собственный цвет фона и собственную функцию окна. Добавить в окно приложения горизонтальную полосу прокрутки. Включить стиль WS_HSCROLL в описание стиля создаваемого функцией CreateWindow окна
29. Создать простейшее приложение WinAPI, содержащее обработку сообщения WM_PAINT для вывода текста и геометрических фигур в окно.
30. Создать простейшее приложение WinAPI, обрабатывающее сообщения от таймера.
31. Создать простейшее приложение WinAPI, с использованием элементов управления edit, text, label, button.

3.2. Задания для оценки освоения МДК.01.02

3.2.1. Распределение типов контрольных заданий по элементам знаний и умений

	У1	У2	У3	У4	У5	З1	З2	З3	З4
Тема 1. Основные возможности комплексной среды разработки приложений Qt									
Тема 1.1Технология программирования на Qt. Установка среды Qt на ПК пользователя.	ПЗ № 8,24,25 ЛР № 1,2 ВСР № 1,2	ПЗ № 8,24,25 ЛР № 1,2 ВСР № 1,2	ПЗ № 8,24,25 ЛР № 1,2			В № 7-14,44,45 ЛР № 1,2	В № 20,21,25-40 ЛР № 1,2	В № 1-6,16-19,22,23,24,26 ЛР № 1,2 ВСР № 1,2	В № 49,50
Тема 1.2. Создание диалоговых окон	В №52 ПЗ № 10,12 ЛР № 4 ВСР №4	ПЗ № 10,12 ЛР № 4,15	ПЗ № 10,12 ЛР № 4			В № 47,53,54 ЛР № 4,15	ЛР № 4	В № 48 ЛР № 4 ВСР №4	
Тема 1.3 Создание главных окон	ПЗ № 8,26 ЛР № 3	ПЗ № 8 ЛР № 3	ПЗ № 8 ЛР № 3			В № 52,55 ЛР № 3	В № 52,55 ЛР № 3	В № 57 ЛР № 3 ВСР 3	
Тема 1.4 Реализация функциональности приложения	ПЗ № 7,27,28 ЛР № 5,14	ПЗ № 7,28 ЛР № 5,14	ПЗ № 7,28 ЛР № 5,14			В № 52,56 ЛР № 5,14	В № 52,56 ЛР № 5,14	В № 57 ЛР № 5,14 ВСР №5б14	
Тема 1.5. Создание пользовательских виджетов	ПЗ № 9,11,28 ЛР № 6,13	ПЗ № 9,11 ЛР № 6,13	ПЗ № 9,11 ЛР № 6,13			В№ 58,59,60 ЛР № 6,13	В№ 58,59,60 ЛР № 6,13	В № 57 ЛР № 6,13 ВСР №6,13	
Тема 2. Средний уровень Qt программирования									
Тема 2.1 Управление компоновкой	ПЗ № 13,14,15,16 ЛР № 6,11	ПЗ № 13,14,15,16 ЛР № 6,11	ПЗ № 13,14,15 16,17,18 ЛР № 6,11			В № 53,54 ЛР № 6,11	В № 53,54 ЛР № 6,11	ЛР № 6,11 ВСР №6,11	
Тема 2.2 Обработка событий	ПЗ № 19,20,21 ЛР № 6	ПЗ № 19,20,21 ЛР № 6	ПЗ № 19,20,21 ЛР № 6,17			В № 62,63 ЛР № 6	В № 62,63 ЛР № 6	В № 57 ЛР № 6,17 ВСР №6,17	
Тема 2.3.Графика 2D и 3D.	ПЗ № 23,30 ЛР № 7	ПЗ № 23 ЛР № 7	ПЗ № 23 ЛР № 7,18,19			В № 58 ЛР № 7	В № 58 ЛР № 7	В № 57 ЛР № 7,18,19 ВСР №7,18,19	
Тема 2.4 Технология “drag-and-drop”	ПЗ № 5,6 ЛР № 8	ПЗ № 5,6 ЛР № 8	ЛР № 8			В № 56 ЛР № 8	ЛР № 8	В № 59 ЛР № 8 ВСР №8	
Тема 2.5 Классы отображения элементов	ПЗ № 1,2,29 ЛР № 9	ПЗ №1,2 ЛР № 9	ЛР № 9			В № 61 ЛР № 9	В № 61 ЛР № 9	В № 57 ЛР № 9 ВСР №9	
Тема 2.6 Классы-контейнеры	ПЗ № 3,4 ЛР № 10	ПЗ № 3,4 ЛР № 10	ЛР № 10			ЛР № 10	В № 51 ЛР № 10	ЛР № 10 ВСР №10	
Тема 2.7 Ввод-вывод	ПЗ № 20,21 ЛР № 12	ПЗ ,21 ЛР № 12	ПЗ № 20,21 ЛР № 12	ЛР № 20,21 ВСР №2	ЛР № 20,21	В № 63 ЛР № 12	В № 63 ЛР № 12	ЛР № 12 ВСР №12	В № 64 ЛР № 20,21

ПЗ – практическое задание
ЛР – лабораторная работа
ВСР – внеаудиторная самостоятельная работа
В – вопрос

3.2.2 Лабораторные работы к МДК 01.02

Лабораторная работа № 1

Тема: Создание простого приложения **Qt** с применением объекта типа QApplication, который обеспечивает управление всеми ресурсами приложения, а также обработку событий, поступающих от операционной системы.

Цель: разработать код программного модуля с использованием элемент типа QSpinBox, слайдера – элемент типа QSlider.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.

За не полностью выполненную работу выставляется – 1 балл.

За невыполненную работу выставляется – 0 баллов.

Методические указания:

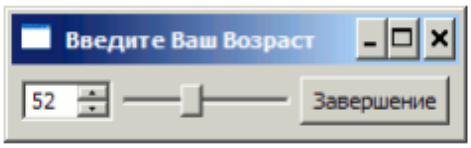
На рисунке показан внешний вид простого приложения, в котором предлагается ввести возраст с использованием одного из трех вариантов ввода:

1) непосредственного ввода числа,

2) посредством стрелок (элемент типа QSpinBox), последовательно увеличивающих или уменьшающих значение,

3) с помощью специального ползунка (слайдера – элемент типа QSlider).

Кроме того, задано верхнее ограничение вводимого возраста, что должно быть корректно отработано слайдером в крайних положениях, а изменение значения любым способом должно синхронизировать его положение.



Ход работы:

Создадим директорию ex1, в которую запишем файл ex1.cpp, содержащий следующий код. Для этого можно воспользоваться любым текстовым редактором, например Блокнот или Notepad.

```
#include <QApplication>
#include <QHBoxLayout>
#include <QVBoxLayout>
#include <QSlider>
#include <QSpinBox>
#include <QPushButton>
#include <QTextCodec>
// Преобразуем входную последовательность символов в кодировку UNICODE
#define RUS( str ) codec->toUnicode(str)
int main(int argc, char *argv[])
{
  QApplication app(argc, argv);
  // Обеспечиваем перекодировку русских букв из кодировки,
  // в которой написана программа.
  // "Windows-1251" - для Windows, "KOI8-R" - Linux и т.д.
  QTextCodec * codec = QTextCodec::codecForName("Windows-1251");

// Создаем главное окно
  QWidget *hbox = new QWidget();
  hbox->setWindowTitle( RUS("Введите Ваш Возраст") );
  QSpinBox *spinBox = new QSpinBox( hbox );
  QSlider *slider = new QSlider(Qt::Horizontal, hbox );
  spinBox->setRange(0, 130);
  slider->setRange(0, 130);
  spinBox->setValue(35);
  QPushButton * btn = new QPushButton( RUS("Завершение"), hbox );
//*************************************************************
  QHBoxLayout *layout = new QHBoxLayout; // выравнивание по горизонтали
  //QVBoxLayout *layout = new QVBoxLayout; // выравнивание по вертикали
  layout->setContentsMargins(5,5,5,5);  // устанавливаем внешние границы
  layout->setSpacing(5); // устанавливаем интервал элементов внутри
  hbox->setLayout(layout);  // связываем layout с hbox
  // устанавливаем порядок следования элементов
  layout->addWidget(spinBox);
  layout->addWidget(slider);
  layout->addWidget(btn);
//*************************************************************
  // связываем сигнал изменения spinBox со слотом slider
  QObject::connect(spinBox, SIGNAL(valueChanged(int)),
    slider, SLOT(setValue(int)));
  // связываем сигнал изменения slider со слотом spinBox
  QObject::connect(slider, SIGNAL(valueChanged(int)),
    spinBox, SLOT(setValue(int)));
  // связываем сигнал нажатия btn со слотом close главного окна
  QObject::connect(btn, SIGNAL(clicked(bool)),
    hbox, SLOT(close()));
  hbox->show();           // отображаем окно
  return app.exec();      // запускаем цикл обработки сообщений
}
```

В приведенной программе объект типа QApplication обеспечивает управление всеми ресурсами приложения, а также обработку событий, поступающих от операционной системы.

По терминологии Qt, визуальные объекты – элементы графического интерфейса пользователя называются виджетами (widget – window gadget) и являются потомками класса QWidget. Любой из них может стать главным окном или быть использован другим виджетом, выполняющим роль контейнера. В представленном выше примере виджет hbox назначается главным окном, а его отображение обеспечивается вызовом функции hbox->show(). В главном окне (виджете) размещено три подчиненных виджета – объекты spinBox и slider типов QSpinBox и QSlider, а также btn типа QPushButton (кнопка). Для каждого виджета может быть выбрана одна схема размещения подчиненных виджетов (Layout). В частности для hbox используем схему горизонтального выравнивания QHBoxLayout * layout. Связывание главного виджета со схемой размещения выполняется вызовом

hbox->setLayout(layout). Подчиненные виджеты будут отображены в порядке их добавления (вызовы layout->addWidget).

Следующие вызовы обеспечат связывание сигналов valueChanged() объектов spinBox, slider таким образом, чтобы изменение любого из них приводило к изменению другого.

```
QObject::connect(spinBox, SIGNAL(valueChanged(int)), slider, SLOT(setValue(int)));
QObject::connect(slider, SIGNAL(valueChanged(int)), spinBox, SLOT(setValue(int)));
```

Связывание сигнала clicked() объекта btn со слотом close() главного окна приведет к тому, что по нажатию на эту кнопку, окно hbox будет закрыто.

```
QObject::connect(btn, SIGNAL(clicked(bool)), hbox, SLOT(close()));
```

Компиляция и компоновка проекта.

После того, как cpp-файл с текстом программы создан, необходимо создать qmake проект для её сборки. Для использования средств разработки, необходимо, чтобы переменные среды окружения, указывающие местоположение и тип компилятора, который будет использован при сборке Qt-проектов, были правильно определены.

.

Для создания файла-проекта, включающего файлы текущей директории, воспользуемся специальной командой qmake -project.

В текущей директории должен появиться файл ex1.pro. Содержимое этого файла определяет характеристики процесса сборки исполняемого файла из файлов проекта.

Файл отредактируем следующим образом:

```
TEMPLATE = app          # тип исполняемого файла - .exe
TARGET = ex1            # имя исполняемого файла – ex1
QT += gui
CONFIG += release       # имя поддиректории для исполняемого файла
# Input
SOURCES = ex1.cpp       # имя исходного файла программы
```

Запускаем qmake ex1.pro, который на базе файла ex1.pro сформирует файл Makefile, определяющий фактический порядок сборки программы, положение компилятора и необходимых библиотек.

После того, как в текущей директории появился файл Makefile, вводим команду nmake (mingw32-make или make в зависимости от компилятора и при их наличии в текущей версии Qt).

Результат сборки программы – файл ex1.exe.

В процессе работы ex1.exe могут потребоваться динамические библиотеки QtCore4.dll и QtGui4.dll, которые должны находиться в путях автовызова, устанавливаемых системной переменной Path, или быть скопированы в директорию запуска приложения ex1.exe.

Задание: Замените в программе схему выравнивания QHBoxLayout на QVBoxLayout и зафиксируйте результат.

Лабораторная работа № 2

Тема: Создание простого приложения в QtDesigner

Цель: разработка кода программного модуля, созданного в QtDesigner, позволяющего использовать менеджеры компоновки.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.

За не полностью выполненную работу выставляется – 1 балл.

За невыполненную работу выставляется – 0 баллов.

Методические указания:

Рассмотрим разработку прототипов диалогов в QtDesigner.

QtDesigner, помимо разнообразных виджетов, позволяет использовать менеджеры компоновки.

В MS Windows исполняемый файл QtDesigner называется designer.exe и может быть запущен из консоли «Qt Command Prompt» по своему имени designer или через меню Пуск/Qt SDK by Nokia v.../Tools/Qt Designer. Qt Designer входит в любой комплект

Qt, однако, в комплекте Qt SDK может не иметь ярлыка, доступного из меню «Пуск», так как его функции в Qt SDK дублирует QtCreator. В этом случае можно непосредственно запустить программу c:\Qt\xx\qt\bin\designer.exe.

Визуальное проектирование формы приложения

После запуска QtDesigner появится окно, показанное на рисунке 6. В качестве примера приложения создадим диалог без кнопок по шаблону «Dialog without Buttons».

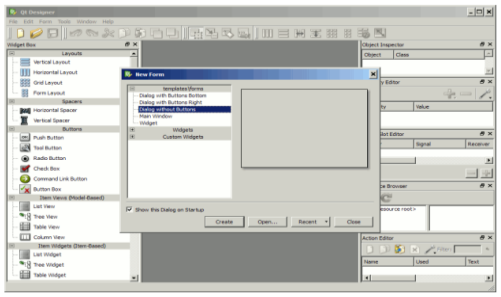


Рисунок 6 – Выбор шаблона формы в QtDesigner

После выбора шаблона появится форма, показанная на рисунке 7. Изменим название диалога посредством редактирования свойства objectName=DialogEx2 в правой колонке.

Лабораторная работа № 3
Тема: визуальный компонент QLabel
Цель: Создать код программы с визуальным компонентом QLabel, который отображает надпись "Hello, Qt!".

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Ниже приводится текст простейшей Qt программы:

```
1 #include <qapplication.h>
2 #include <qlabel.h>
3 int main(int argc, char *argv[])
4 {
5     QApplication app(argc, argv);
6     QLabel *label = new QLabel("Hello, Qt!", 0);
7     app.setMainWidget(label);
```

```
 8     label->show();
 9     return app.exec();
10 }
```

Здесь, в строках 1 и 2, подключаются определения классов QApplication и QLabel.

В строке 5 создается экземпляр класса QApplication, который управляет ресурсами приложения. Конструктору QApplication передаются аргументы argc и argv, поскольку Qt имеет возможность обрабатывать аргументы командной строки.

В строке 6 создается визуальный компонент QLabel, который отображает надпись "Hello, Qt!". В терминологии Qt, все визуальные компоненты, из которых строится графический интерфейс, называются *виджетами* (widgets). Кнопки, меню, полосы прокрутки и разнообразные рамки -- все это виджеты. Одни виджеты могут содержать в себе другие виджеты, например, главное окно приложения -- это самый обычный виджет, который может содержать QMenuBar, QToolBar, QStatusBar и др. Аргумент 0, передаваемый конструктору QLabel (в строке 6) -- это "пустой" (null) указатель, который сообщает о том, что этот виджет не имеет "хозяина", т.е. не включается в другой виджет.

В строке 7 назначается "главный" виджет приложения. Когда пользователь закрывает "главный" виджет приложения (например, нажатием на кнопку "X" в заголовке окна), то программа завершает свою работу. Если в программе не назначить главный виджет, то она продолжит исполнение в фоновом режиме даже после того, как пользователь закроет окно.

В строке 8, метка делается видимой. Виджеты всегда создаются невидимыми, чтобы у программиста оставалась возможность настроить параметры отображения до того, как они станут видимы.

В строке 9 выполняется передача управления библиотеке Qt. С этого момента программа переходит в режим ожидания, когда она ничего не делает, а просто ждет действий пользователя, например, нажатие на клавишу или кнопку мыши.

Любое действие пользователя порождает *событие*, в ответ на которое программа может вызвать одну или более функций. В этом смысле, приложения с графическим интерфейсом кардинально отличаются от обычных программ, с пакетной обработкой данных, которые приняв ввод от пользователя, они самостоятельно обрабатывают его, выдают результаты и завершают свою работу без дальнейшего участия человека.

Теперь самое время проверить работу приложения. Скопируйте текст программы в файл, с именем hello.cpp, в каталог hello.

Перейдите в этот каталог и дайте команду:

```
qmake -project
```

она создаст платформо-независимый файл проекта (hello.pro), а затем дайте следующую команду:

```
qmake hello.pro
```

Эта команда создаст Makefile, на основе файла проекта. Дайте команду make, чтобы скомпилировать программу и затем запустите ее, набрав в командной строке **hello** (в Windows) или **./hello** (в Unix) или **open hello.app** (в Mac OS X). Если вы работаете в Windows и используете Microsoft Visual C++, то вместо команды make вы должны дать команду nmake. Как альтернативный вариант -- вы можете создать проект Visual Studio из файла hello.pro, запустив команду:

```
qmake -tp vc hello.pro
```

и затем скомпилировать программу в Visual Studio.

Лабораторно-практическое задание № 4
Тема: Создание дочернего класса от QDialog
Цель: разработка кода программного модуля, использующего создание дочернего класса от QDialog.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Диалог будет реализован в виде класса, со своей собственной функциональностью. Это будет независимый компонент со своими сигналами и слотами.

Исходные тексты приложения будут размещаться в двух файлах: finddialog.h и finddialog.cpp. Начнем с файла finddialog.h.

```
1 #ifndef FINDDIALOG_H
```

```
 2 #define FINDDIALOG_H
 3 #include <qdialog.h>
 4 class QCheckBox;
 5 class QLabel;
 6 class QLineEdit;
 7 class QPushButton;
```

Строки 1 и 2 (и 27) предотвращают множественное подключение файла-заголовка.

В строке 3 подключается определение класса QDialog -- базового для всех диалогов в Qt. QDialog порожден от класса QWidget.

Строки с 4 по 7 -- это опережающие описания классов Qt, которые используются в нашем диалоге. **Опережающее описание** сообщает компилятору C++ о том, что этот класс существует, но подробности описания класса (обычно расположенного в отдельном заголовочном файле) здесь использоваться не будут. Ниже мы еще вернемся к этой теме.

Теперь определим класс FindDialog, указав в качестве родительского класса QDialog:

```
 8 class FindDialog : public QDialog
 9 {
10     Q_OBJECT
11 public:
12     FindDialog(QWidget *parent = 0, const char *name = 0);
```

Определение класса начинается с вызова макроса **Q_OBJECT**. Это обязательное требование для всех классов, которые определяют свои собственные сигналы и слоты.

Далее следует типичный, для всех виджетов в Qt, конструктор -- FindDialog. Параметр **parent** задает "владельца", т.е. виджет, на котором будет размещен данный компонент, а параметр **name** -- имя (название) виджета. Параметр **name** не является обязательным, в основном он используется для нужд отладки и тестирования.

```
13 signals:
14     void findNext(const QString &str, bool caseSensitive);
15     void findPrev(const QString &str, bool caseSensitive);
```

В секции **signals** описаны два сигнала, которые может посылать наше диалоговое окно, при нажатии на кнопку Find. Если включен флажок "Search backward" (поиск в обратном направлении), то посылается сигнал findPrev(), иначе -- findNext().

Ключевое слово **signals**, фактически является макроопределением. Препроцессор C++ преобразует его в стандартное представление C++ и только потом оно будет передано компилятору.

```
16 private slots:
17     void findClicked();
18     void enableFindButton(const QString &text);
19 private:
20     QLabel *label;
21     QLineEdit *lineEdit;
22     QCheckBox *caseCheckBox;
23     QCheckBox *backwardCheckBox;
24     QPushButton *findButton;
25     QPushButton *closeButton;
26 };
27 #endif
```

В приватной секции класса мы объявили два слота. Они необходимы для обеспечения взаимодействия с подчиненными виджетами, указатели на которые описаны чуть ниже. Ключевое слово **slots**, так же как и **signals**, является макроопределением.

Поскольку все поля-переменные, - это указатели, нам нет нужды подключать заголовочные файлы, содержащие полные определения этих классов. Благодаря наличию опережающего описания, компилятор удовольствуется тем, что есть. Вместо опережающего описания классов мы могли бы подключить соответствующие заголовочные файлы (<qcheckbox.h>, <qlabel.h>), но это отрицательно скажется на скорости компиляции. Для маленьких приложений это не так заметно, но для больших проектов опережающее описание может дать существенный выигрыш.

Перейдем к файлу finddialog.cpp, который содержит реализацию класса FindDialog:

```
1 #include <qcheckbox.h>
2 #include <qlabel.h>
3 #include <qlayout.h>
4 #include <qlineedit.h>
5 #include <qpushbutton.h>

6 #include "finddialog.h"
```

Здесь подключаются заголовочные файлы с описаниями используемых классов Qt. В строке 6 подключается определение нашего класса. Для большинства классов Qt, их определения находятся в заголовочных файлах, имена которых повторяют имя класса (все символы в именах файлов переводятся в нижний регистр) и дополняются символами .h.

```
 7 FindDialog::FindDialog(QWidget *parent, const char *name)
 8     : QDialog(parent, name)
 9 {
10     setCaption(tr("Find"));
11     label = new QLabel(tr("Find &what:"), this);
12     lineEdit = new QLineEdit(this);
13     label->setBuddy(lineEdit);
14     caseCheckBox = new QCheckBox(tr("Match &case"), this);
15     backwardCheckBox = new QCheckBox(tr("Search &backward"), this);
16     findButton = new QPushButton(tr("&Find"), this);
17     findButton->setDefault(true);
18     findButton->setEnabled(false);
19     closeButton = new QPushButton(tr("Close"), this);
```

В строке 8, конструктору базового класса передаются параметры **parent** и **name**.

В строке 10 задается надпись, которая будет выводиться в заголовке окна -- "Find". Функция tr() определена в классе QObject и любом другом подклассе, описание которого содержит вызов макроса **Q_OBJECT**. Она выполняет трансляцию текста, передаваемого ей, на другие языки человеческого общения. Считается хорошим тоном, все строки, которые будут выводиться на экран, передавать через эту функцию, даже в том случае, если вы не планируете интернационализацию своего приложения. Затем, начиная со строки 11, создаются подчиненные виджеты. Здесь символ амперсанда ('&') используется для обозначения клавиши быстрого доступа (акселератор). Например, в строке 16 создается кнопка Find, активировать которую можно нажатием на комбинацию клавиш Alt+F. Амперсанды так же могут использоваться для передачи фокуса ввода: в строке 11 создается метка, с акселератором (Alt+W), а в строке 13 метке назначается "дружественный" (buddy) компонент -- однострочное поле ввода. **"Дружественный"** виджет -- это такой компонент, который будет получать фокус ввода при нажатии ускоряющей комбинации клавиш метки. Таким образом, когда

пользователь нажмет комбинацию клавиш Alt+W (акселератор метки), то фокус ввода будет передан "дружественному" виджету, т.е. -- полю ввода.

В строке 17, вызовом метода setDefault(true) [5], назначается кнопка по-умолчанию. Кнопка по-умолчанию -- это такая кнопка, которая будет активироваться при нажатии на клавишу Enter. В строке 18 накладывается запрет на кнопку Find. Когда виджет запрещен, он обычно отображается в серых тонах и не реагирует на действия пользователя.

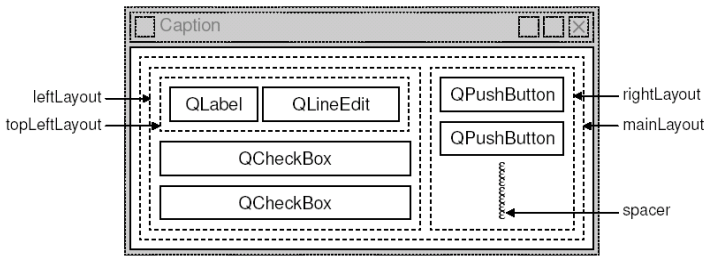
```
20 connect(lineEdit, SIGNAL(textChanged(const QString &)),
21     this, SLOT(enableFindButton(const QString &)));
22 connect(findButton, SIGNAL(clicked()),
23     this, SLOT(findClicked()));
24 connect(closeButton, SIGNAL(clicked()),
25     this, SLOT(close()));
```

Приватный слот enableFindButton(const QString &) вызывается при изменении содержимого поля ввода. Приватный слот findClicked()вызывается, когда пользователь щелкает по кнопке Find. Работа приложения завершается после щелчка по кнопке Close. Слот close()наследуется от класса QWidget и его поведение по-умолчанию -- сокрытие виджета. Реализация слотов findClicked() иenableFindButton(const QString &) будет приведена ниже.

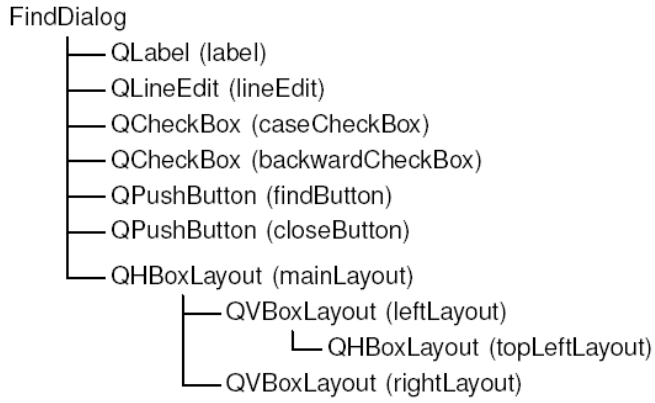
Поскольку QObject является одним из предков класса FindDialog, то мы можем опустить префикс QObject:: перед именем метода connect().

```
26     QHBoxLayout *topLeftLayout = new QHBoxLayout;
27     topLeftLayout->addWidget(label);
28     topLeftLayout->addWidget(lineEdit);
29     QVBoxLayout *leftLayout = new QVBoxLayout;
30     leftLayout->addLayout(topLeftLayout);
31     leftLayout->addWidget(caseCheckBox);
32     leftLayout->addWidget(backwardCheckBox);
33     QVBoxLayout *rightLayout = new QVBoxLayout;
34     rightLayout->addWidget(findButton);
35     rightLayout->addWidget(closeButton);
36     rightLayout->addStretch(1);
37     QHBoxLayout *mainLayout = new QHBoxLayout(this);
38     mainLayout->setMargin(11);
39     mainLayout->setSpacing(6);
40     mainLayout->addLayout(leftLayout);
41     mainLayout->addLayout(rightLayout);
42 }
```

На заключительном этапе выполняется выравнивание виджетов с помощью **менеджеров размещения**. Менеджер размещения (layout manager) -- это объект, который управляет размерами и положением виджетов. Qt предоставляет в наше распоряжение три менеджера размещения:QHBoxLayout выравнивает виджеты по горизонтали, QVBoxLayout -- по вертикали и QGridLayouts -- по сетке. Менеджеры размещения (или, если хотите, менеджеры компоновки) могут содержать как отдельные виджеты, так и другие менеджеры размещения. Вкладывая друг в другаQHBoxLayout, QVBoxLayout и QGridLayouts, в различных комбинациях, можно выстроить весьма замысловатый интерфейс диалога.



В приложении мы использовали два QHBoxLayout и два QVBoxLayout, Внешний менеджер компоновки (mainLayout) является главным, поскольку при создании ему был назначен, в качестве владельца, экземпляр классаFindDialog -- **(this)**. Он отвечает за размещение всех визуальных компонентов в области окна приложения. Оставшиеся 3 менеджера размещения являются подчиненными. Маленькая "пружинка", которая видна в правом нижнем углу рисунка -- это **распорка** (spacer). Она заполняет пустое пространство под кнопками Find и Close, заставляя их держаться в верхней части области выравнивания.
Здесь есть один важный момент. Менеджеры компоновки не являются виджетами. Они порождены от класса QLayout, который в свою очередь порожден от класса QObject. На рисунке, контуры виджетов отрисованы сплошными линиями, а областей компоновки -- пунктирными, чтобы подчеркнуть различия, имеющиеся между ними. Во время работы приложения, менеджеры размещения (области выравнивания) не видны.
Хотя менеджеры компоновки и не являются виджетами (визуальными компонентами), тем не менее они могут иметь как владельца, так и подчиненные компоненты. Понятия термина "владелец", у менеджера размещения и виджета, различаются. Если менеджер размещения встраивается в виджет (который передается менеджеру в качестве владельца), как это происходит в случае с **mainLayout**, то он автоматически встраивается в этот виджет. Если менеджер создается без владельца (в данном случае это: **topLeftLayout**, **leftLayout** и**rightLayout**), то он должен быть включен в состав другого менеджера, вызовом addLayout().
Механизм "владелец-подчиненный" реализован в классе QObject, который является предком как для QWidget, так и для QLayout. Когда создается некий объект (виджет, менеджер компоновки или что-то еще), для которого явно указывается владелец, то он добавляется владельцем в свой список подчиненных компонентов. Когда владелец уничтожается, он проходит по списку подчиненных компонентов и уничтожает их одного за другим. Подчиненные компоненты, в свою очередь просматривают свои списки и уничтожают компоненты, подчиненные им и так до тех пор, пока дело не дойдет до компонентов, которые не имеют подчиненных объектов.
Этот механизм упрощает управление памятью в приложении, снижая риск "утечки". Единственные объекты, которые необходимо уничтожать явно -- это те, которые были созданы оператором new, и не имеют владельца. Если первым удаляется подчиненный компонент, то Qt автоматически исключит его из списка владельца.
Владелец имеет особое значение для виджетов. Подчиненные виджеты отображаются на экране внутри области, принадлежащей его владельцу. Когда удаляется какой-либо владелец, он не только удалит подчиненные объекты из памяти, но и сотрет их на экране.
Когда один менеджер размещения вставляется в другой, с помощью функции addLayout(), то вложенный менеджер автоматически делается подчиненным объемлющему. В противоположность этому, когда в менеджер размещения вставляется виджет, вызовомaddWidget(), то последний не меняет своего владельца.
На рисунке показано дерево "владелец-подчиненный" для приложения Find. Порядок взаимоотношений в этом дереве легко выводится из текста конструктора FindDialog, достаточно выделить строки, содержащие new и addLayout(). Важное замечание: Запомните, менеджеры размещения НЕ ЯВЛЯЮТСЯ владельцами виджетов, размещением которых они управляют.



В дополнение к менеджарам размещения, Qt предоставляет несколько **виджетов размещения**: QHBox, QVBox и QGrid. Для своих подчиненных компонентов, эти классы выступают как в качестве владельцев так и в качестве менеджеров размещения. Для небольших приложений виджеты размещения более удобны, но они менее гибкие и требуют больший объем ресурсов, чем менеджеры.

На этом мы заканчиваем рассмотрение реализации конструктора FindDialog. Поскольку подчиненные виджеты и менеджеры размещения создавались нами с помощью оператора new, то, казалось бы, возникает необходимость в написании деструктора, который удалил бы из памяти все, созданные нами компоненты, во время завершения приложения. Однако в этом нет необходимости, поскольку Qt автоматически сделает это во время уничтожения владельца, т.е. во время уничтожения класса FindDialog.

Перейдем к рассмотрению слотов:

```
43 void FindDialog::findClicked()
44 {
45     QString text = lineEdit->text();
46     bool caseSensitive = caseCheckBox->isOn();

47     if (backwardCheckBox->isOn())
48         emit findPrev(text, caseSensitive);
49     else
50         emit findNext(text, caseSensitive);
51 }

52 void FindDialog::enableFindButton(const QString &text)
53 {
54     findButton->setEnabled(!text.isEmpty());
55 }
```

Слот findClicked() вызывается всякий раз, когда пользователь щелкает мышкой по кнопке Find. Кнопка выдает сигнал findPrev() илиfindNext(), в зависимости от состояния флажка Search backward. Ключевое слово **emit** является макросом, определенным в библиотеке Qt.

Слот enableFindButton() вызывается, когда пользователь изменяет содержимое поля ввода. Если оно содержит какие-либо символы, то разрешается кнопка Find, в противном случае она запрещается.

Реализацией этих двух слотов завершается разработка нашего диалога. Теперь создадим файл main.cpp, в котором разместим текст основной программы для тестирования нашего виджета:

```
1 #include <qapplication.h>

2 #include "finddialog.h"
```

```
 3 int main(int argc, char *argv[])
 4 {
 5     QApplication app(argc, argv);
 6     FindDialog *dialog = new FindDialog;
 7     app.setMainWidget(dialog);
 8     dialog->show();
 9     return app.exec();
10 }
```

дадим команду qmake, как обычно. На этот раз, поскольку наш класс FindDialog содержит вызов макроопределения **Q_OBJECT**, утилита qmakeвключит в Makefile правила, вызывающие утилиту moc -- компилятор метаобъектов.

Для корректной работы утилиты moc необходимо, чтобы описание класса размещалось в заголовочном файле, отдельно от файла с реализацией. Код, сгенерированный утилитой moc подключает этот заголовочный файл.

Классы, в определении которых встречается макрос **Q_OBJECT**, должны обрабатываться компилятором метаобъектов в обязательном порядке. На самом деле, это не такая большая проблема, поскольку qmake автоматически добавит все необходимые правила в Makefile. Но если вы забудете перегенерировать Makefile, то линковщик будет "жаловаться" на отсутствие некоторых функций. Эти сообщения об ошибках могут порой вводить в заблуждение. Например, GCC выдает примерно такое предупреждение:

```
finddialog.o(.text+0x28): undefined reference to
FindDialog::QPaintDevice virtual table
```

Visual C++ такое:

```
finddialog.obj : error LNK2001: unresolved external symbol
"public:~virtual bool __thiscall FindDialog::qt_property(int,
int,class QVariant *)"
```

Если это произошло, то перезапустите qmake еще раз, чтобы обновить Makefile, а затем пересоберите приложение.

Теперь запустите программу. Проверьте работу акселераторов Alt+W, Alt+C, Alt+B и Alt+F. Попробуйте "пройтись" по виджетам с помощью клавиши Tab. По-умолчанию, порядок навигации с помощью клавиши Tab, соответствует порядку, в котором создавались компоненты. Но он может быть изменен вызовом метода QWidget::setTabOrder(). Установку акселераторов и настройку порядка навигации по компонентам, с помощью клавиши Tab, можно считать дружественным жестом в сторону пользователей, которые не могут или не хотят пользоваться мышью. Удобное управление с клавиатуры высоко оценят опытные пользователи.

Мы будем использовать наш диалог в реально работающем приложении. Там сигналы findPrev() и findNext() будут подключаться к соответствующим слотам.

Лабораторная работа № 5

Тема: Быстрая разработка диалогов.

Цель: разработка кода программного модуля с помощью **Qt Designer** .

Время выполнения: 80 минут

Проверяемые результаты обучения

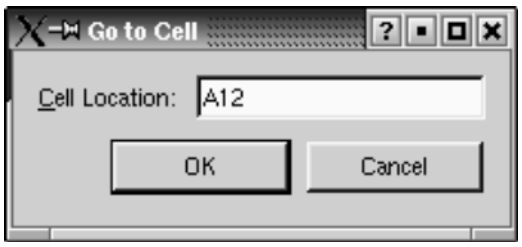
Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во**

		баллов)
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

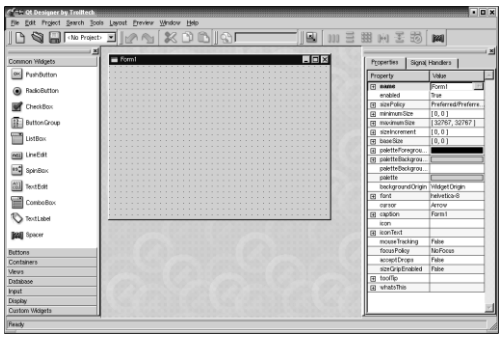
За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Qt разрабатывалась так, чтобы можно было писать код программы вручную без особого напряжения. Тем не менее, **Qt Designer** еще больше расширяет возможности программиста, предоставляя ему возможность визуального дизайна.
С помощью **Qt Designer**, напишем диалог Go-to-Cell ("Перейти к ячейке"), показанный на рисунке ниже. Совершенно неважно, как разрабатывается диалог - вручную ли, или с помощью **Qt Designer**, всегда выполняется одна и та же последовательность действий:

- Создаются и инициализируются подчиненные виджеты.
- Подчиненные виджеты вставляются в менеджеры размещения.
- Настраивается порядок навигации по виджетам клавишей Tab.
- Устанавливаются соединения сигнал-слот
- Реализуются дополнительные слоты диалога, если это необходимо.

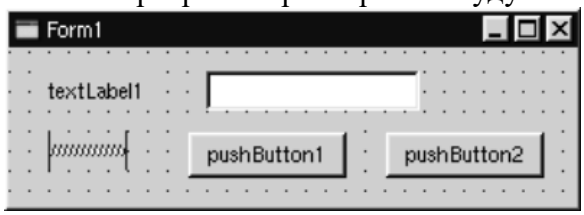


Чтобы запустить **Qt Designer**, выберите пункт Qt 3.2.x|Qt Designer в меню "Пуск" - в ОС Windows. После того, как программа запустится, она предложит на выбор список шаблонов. Щелкните по шаблону **"Dialog"** и нажмите кнопку OK. После этого перед вами должна появиться заготовка будущего окна диалога с именем "Form1".



Для начала разместим виджеты на форме. На инструментальной панели, слева, щелкните по компоненту TextLabel, затем щелкните по форме - в результате на форме появится компонент "метка". Аналогичным образом разместите на форме одно поле ввода (LineEdit), одну горизонтальную распорку (Spacer) и две кнопки (PushButton). Разместите их так, чтобы у вас получилось нечто похожее на рисунке.. Не тратьте слишком много времени на позиционирование виджетов. Мы все равно будем использовать менеджеры размещения, которые выполнят эту работу за нас.

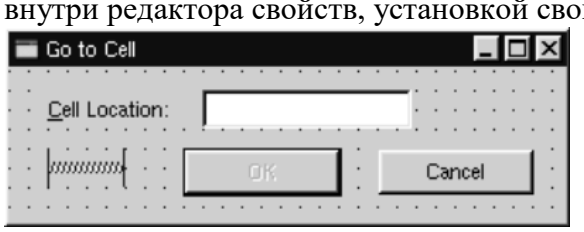
Распорка (spacer) отображается на заготовке в виде синей пружинки. Во время работы уже готовой программы распорки не будут отображаться.



Установите свойства для каждого из виджетов, используя **Редактор свойств**, расположенный в правой части главного окна **Qt Designer**.

1. Щелкните по компоненту TextLabel и запишите в его свойство **name** строку "label", а в свойство **text** -- "&Cell Location:".
2. Щелкните по компоненту LineEdit и запишите в свойство **name** строку "lineEdit".
3. Для распорки запишите в свойство **orientation** "Horizontal".
4. Для первой кнопки запишите в свойство **name** строку "okButton", в свойство **enabled** -- "False", в свойство **default** -- "True" и в свойство **text** -- "OK".
5. Для второй кнопки. Запишите в свойство **name** строку "cancelButton", а в свойство **text** -- в "OK".
6. Щелкните в любом свободном месте формы и запишите в свойство **name** строку "GoToCellDialog", а в свойство **caption** -- "Go to Cell".

Но это еще не все, нам нужно назначить дружественный компонент для метки, который будет реагировать на акселератор Alt+C. На данный момент метка отображается как "&Cell Location:". Выберите пункт меню Tools|Set Buddy (курсор мыши приобретет вид крестика). Затем поместите указатель мыши на метку, нажмите левую кнопку и, удерживая ее в нажатом положении, переместите указатель мыши на компонент LineEdit. Затем отпустите кнопку мыши. Изображение метки изменится -- символ амперсанда исчезнет, а первый символ метки приобретет знак подчеркивания. В принципе, то же самое можно сделать внутри редактора свойств, установкой свойства **buddy** метки.



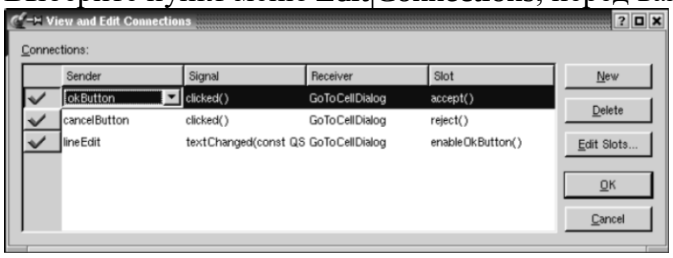
Следующий шаг -- размещение виджетов на форме с помощью менеджеров компоновки:

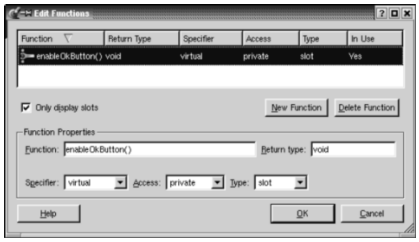
1. Щелкните мышью по метке. Нажмите клавишу Ctrl и удерживая ее -- щелкните по полю ввода. Оба компонента окажутся выделенными. Теперь выберите пункт меню Layout|Lay Out Horizontally.
2. Щелкните мышью по распорке. Нажмите клавишу Shift и удерживая ее -- щелкните сначала по кнопке "OK", а затем по кнопке "Cancel". Теперь выберите пункт меню Layout|Lay Out Horizontally.
3. Щелкните по свободному пространству на форме и выберите пункт меню Layout|Lay Out Vertically.
4. Выберите пункт меню Layout|Adjust Size.

Красные контуры областей выравнивания, которые сейчас видны на заготовке, во время работы программы отображаться не будут.

Теперь выберите пункт меню Tools|Tab Order. На каждом из виджетов, которые могут принимать фокус, появятся цифры в синих кружочках. Щелчками мыши по компонентам установите желаемый порядок навигации клавишей Tab и нажмите Esc.

На этом дизайн внешнего вида формы можно считать завершенным. Теперь перейдем к настройке функциональной части -- свяжем сигналы и слоты и создадим свой слот. Выберите пункт меню Edit|Connections, перед вами откроется окно редактора связей:





Для начала создадим новый слот: щелкните по кнопке Edit Slots.... Перед вами откроется окно редактора слотов. Создайте приватный слот с именем enableOkButton().

Затем необходимо настроить три соединения. Чтобы создать соединение -- щелкните по кнопке "New" и установите поля Sender, Signal, Receiver и Slot, выбирая требуемые значения из выпадающих списков в каждом из них. У вас должно получиться следующее:

okButton clicked() GoToCellDialog accept()
cancelButton clicked() GoToCellDialog reject()

lineEdit textChanged(const QString &) GoToCellDialog enableOkButton()

Чтобы посмотреть, как будет выглядеть окно диалога во время работы программы -- выберите пункт меню Preview|Preview Form. Проверьте порядок навигации клавишей Tab. Проверьте работу акселератора Alt+C (поле ввода должно получить фокус ввода). Нажмите кнопку Cancel, чтобы закрыть окно.

Сохраните результаты работы в файле gotocelldialog.ui в каталоге gotocell и создайте файл main.cpp, в том же каталоге, с помощью обычного текстового редактора:

```
#include <qapplication.h>
#include "gotocelldialog.h"
int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   GoToCellDialog *dialog = new GoToCellDialog;
   app.setMainWidget(dialog);
   dialog->show();
   return app.exec();
}
```

Создайте файл проекта и Makefile утилитой qmake (qmake -project; qmake gotocell.pro). Утилита qmake сама обнаружит файлgotocelldialog.ui и добавит в Makefile все необходимые правила по созданию gotocelldialog.h и gotocelldialog.cpp. Все .ui файлы преобразуются в код C++ с помощью утилиты uic (User Interface Compiler -- Компилятор Пользовательских Интерфейсов).

Вся прелесть **Qt Designer**-а состоит в том, что вы можете свободно изменять дизайн формы без необходимости вторгаться в исходный код на C++. Когда разработка дизайна ведется в тексте программы (вручную) то это может отнять довольно значительное время. **Qt Designer**сохранит ваши силы и время.

Если теперь запустить программу, то вы заметите:

- Кнопка "OK" всегда остается запрещенной.
- Поле ввода принимает не только те символы, из которых может состоять номер искомой ячейки, но и любые другие.

Мы должны решить эти проблемы.

Щелкните дважды по свободному пространству на форме, чтобы вызвать редактор исходного кода. В окне редактора добавьте следующие строки:

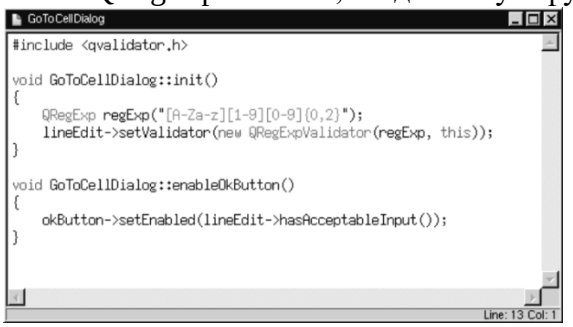
```
#include <qvalidator.h>
void GoToCellDialog::init()
{
   QRegExp regExp("[A-Za-z][1-9][0-9]{0,2}");
   lineEdit->setValidator(new QRegExpValidator(regExp, this));
}
void GoToCellDialog::enableOkButton()
{
   okButton->setEnabled(lineEdit->hasAcceptableInput());
}
```

Функция init() автоматически вызывается конструктором формы (конструктор генерируется утилитой uic). Она настраивает проверку корректности ввода для LineEdit. Qt предоставляет три класса, выполняющих проверку на корректность: QIntValidator, QDoubleValidatorи QRegExpValidator. Для своих нужд мы будем использовать последний, который будет выполнять проверку на основе регулярного выражения: "[A-Za-z][1-9][0-9]{0,2}". Это выражение означает: "Позволить ввод одного алфавитного символа в верхнем

или нижнем регистре, за которым должна следовать одна цифра, в диапазоне от 1 до 9, за которой может следовать до двух цифр, в диапазоне от 0 до 9".

Передавая аргумент **this** (в вызов конструктора QRegExpValidator()), мы делаем объект класса QRegExpValidator подчиненным, по отношению к GoToCellDialog. Таким образом мы снимаем с себя ответственность за удаление этого объекта из памяти по завершении работы приложения.

Слот enableOkButton() разрешает или запрещает кнопку "OK", в зависимости от того, насколько правильный номер ячейки содержится в поле ввода. Для проверки правильности используется функция QLineEdit::hasAcceptableInput(), которая обращается к объекту классаQRegExpValidator, созданному в функции init().



После этого опять сохраните диалог. **Qt Designer** сохранит оба файла -- и gotocelldialog.ui, и gotocelldialog.ui.h. Пересоберите приложение и запустите его. Введите в поле ввода строку "A12" -- кнопка "OK" перейдет в разрешенное состояние. Попробуйте набрать произвольный текст и понаблюдайте за тем, как работает проверка корректности ввода. Нажмите кнопку "Cancel", чтобы завершить работу программы.

В этом примере мы создали диалог с помощью **Qt Designer** и добавили некоторый код, с помощью редактора исходного кода **Qt Designer**-а. Интерфейсная часть диалога была сохранена в файле gotocelldialog.ui (по сути файл формата XML), а исходный текст -- в файлегotocelldialog.ui.h. Это очень удобно, поскольку gotocelldialog.ui.h можно править в любом текстовом редакторе.

Альтернативный подход заключается в разработке формы с помощью **Qt Designer** (как обычно), а затем создается дополнительный класс, порожденный от класса формы, в котором реализуется вся необходимая функциональность. Например, для нашего диалога Go-to-Cell можно было бы создать класс GoToCellDialogImpl, как наследник класса GoToCellDialog и реализовать в нем все необходимые функции. В результате такого подхода, новый заголовочный файл должен получиться таким:

```
#ifndef GOTOCELLDIALOGIMPL_H
#define GOTOCELLDIALOGIMPL_H
#include "gotocelldialog.h"
class GoToCellDialogImpl : public GoToCellDialog
{
   Q_OBJECT
public:
   GoToCellDialogImpl(QWidget *parent = 0, const char *name = 0);
private slots:
   void enableOkButton();
};
#endif
```

А файл с реализацией:

```
#include <qlineedit.h>
#include <qpushbutton.h>
#include <qvalidator.h>
```

```
#include "gotocelldialogimpl.h"
GoToCellDialogImpl::GoToCellDialogImpl(QWidget *parent,
                                       const char *name)
   : GoToCellDialog(parent, name)
{
   QRegExp regExp("[A-Za-z][1-9][0-9]{0,2}");
   lineEdit->setValidator(new QRegExpValidator(regExp, this));
}
void GoToCellDialogImpl::enableOkButton()
{
   okButton->setEnabled(lineEdit->hasAcceptableInput());
}
```

Создание классов-наследников может быть упрощено с помощью утилиты uic и набора дополнительных аргументов командной строки. Так, например, утилита uic, с ключом **-subdecl**, создаст скелетон заголовочного файла, а с ключом **-subimpl** -- соответствующий файл реализации.

В данной книге мы будем работать только с файлами .ui.h, поскольку это наиболее общеупотребимая практика, а создание дочерних классов, с помощью uic, довольно простая задача. Чтобы поглубже разобраться в различиях этих двух подходов, рекомендуем прочитать главу "Designer Approach" в справочном руководстве, поставляемом вместе с **Qt Designer**. Кроме того, прочитайте главу "Creating Dialogs", где показывается, как можно использовать вкладку "Members" для создания полей (переменных-членов) в классе формы.

Лабораторная работа № 6

Тема: Диалоги с изменяющимся внешним видом.

Цель: Разработка кода программного модуля с использованием *Qt Designer для создания диалога.*

Время выполнения: 80 минут

Проверяемые результаты обучения

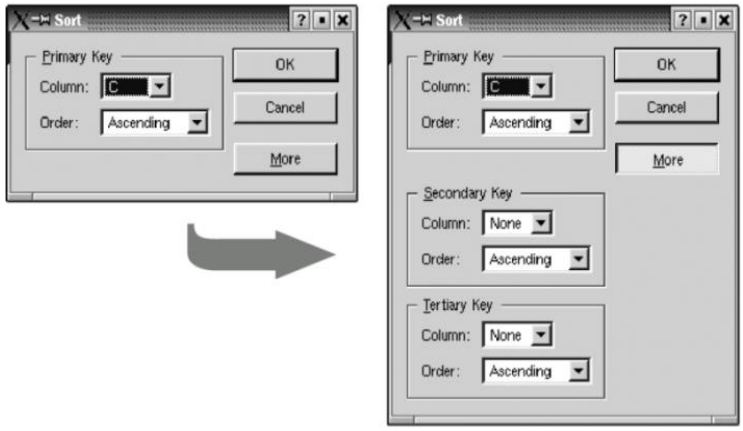
Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	

З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

В некоторых случаях желательно иметь диалоги, которые могут динамически изменять свое представление. Наиболее часто на практике встречаются расширяемые диалоги и многостраничные диалоги. Оба вида диалогов могут быть созданы как в **Qt Designer**, так и в результате ручного кодирования.

Расширяемые диалоги, обычно выводятся на экран в сокращенном варианте, но дают пользователю возможность выбирать между сокращенным и расширенным режимом представления информации. Расширяемые диалоги как правило используются в тех случаях, когда необходимо сокрыть дополнительные сведения, которые не являются обязательными и пользователь явно не выразил свое желание видеть их. В этом разделе мы разберем процесс создания расширяемого диалога, показанного на рисунке ниже с помощью **Qt Designer**.

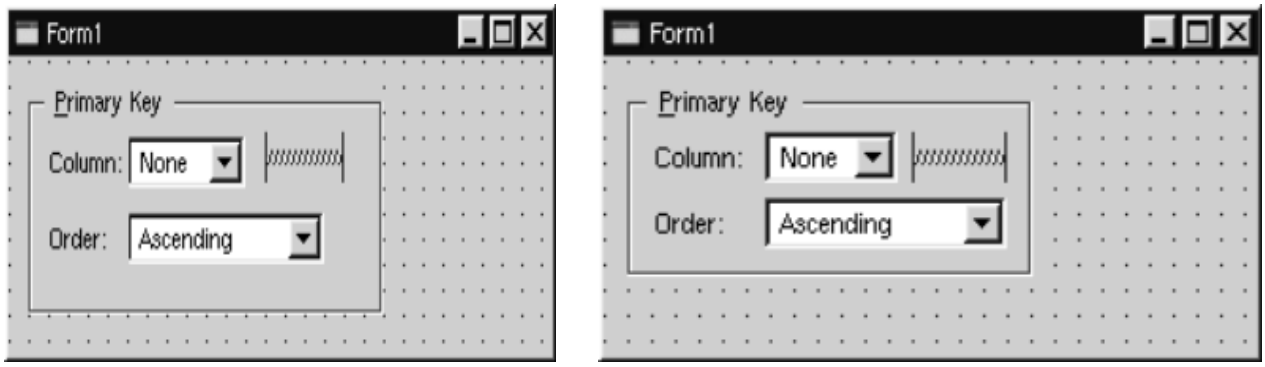


Это диалог сортировки, используемый в электронных таблицах. Он появляется, когда пользователь пытается отсортировать данные по одному или нескольким столбцам. В сокращенном представлении диалог позволяет выбрать столбец и порядок сортировки, в расширенном варианте добавляется возможность указать еще два столбца и порядок сортировки по каждому из них. Кнопка "More" позволяет переходить из сокращенного режима -- в расширенный и обратно.

Мы создадим диалог в его расширенном виде, а потом, во время исполнения, будем скрывать дополнительные виджеты, чтобы обеспечить краткую форму диалога.

1. Положите на заготовку формы GroupBox, два TextLabel, два ComboBox и одну горизонтальную распорку.
2. "Растяните" GroupBox побольше, ухватив мышкой за правый нижний его угол.
3. Разместите виджеты внутри GroupBox-а примерно так, как показано на рисунке ниже.
4. Ухватив мышкой за правый край второго ComboBox-а, сделайте его примерно в два раз больше первого.
5. Запишите в свойство **title**, GroupBox-а, строку "&Primary Key". В свойство **text** первой метки -- "Column:", второй метки -- "Order:".
6. Щелкните мышкой дважды по первому ComboBox. Перед вами появится окно редактора, в котором добавьте один элемент с текстом "None".
7. Щелкните мышкой дважды по второму ComboBox и добавьте элементы "Ascending" и "Descending".

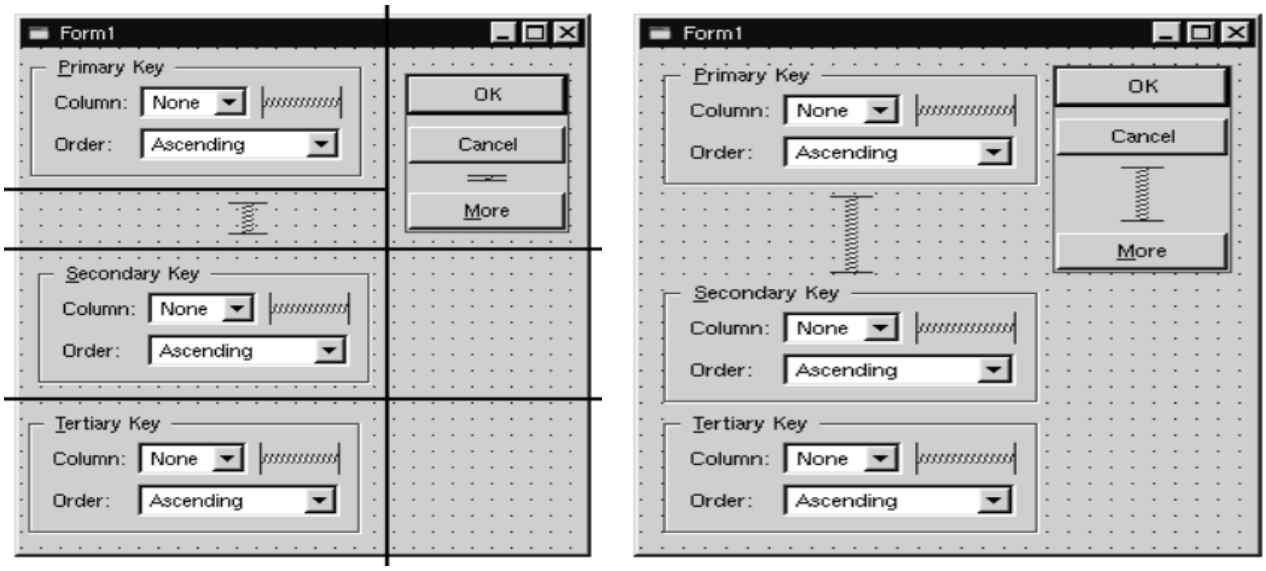
8. Теперь скомпонуем виджеты внутри GroupBox, для этого, щелкните по GroupBox и выберите пункт меню Layout|Lay Out in a Grid. В результате вы должны получить нечто похожее на рисунок.



Если компоновка выполнилась не так как надо или вы допустили какую нибудь ошибку, вы всегда можете выбрать пункт меню Edit|Undo и отменить произведенное действие. После чего можете повторить попытку.

Теперь добавим группы виджетов для расширенного представления:

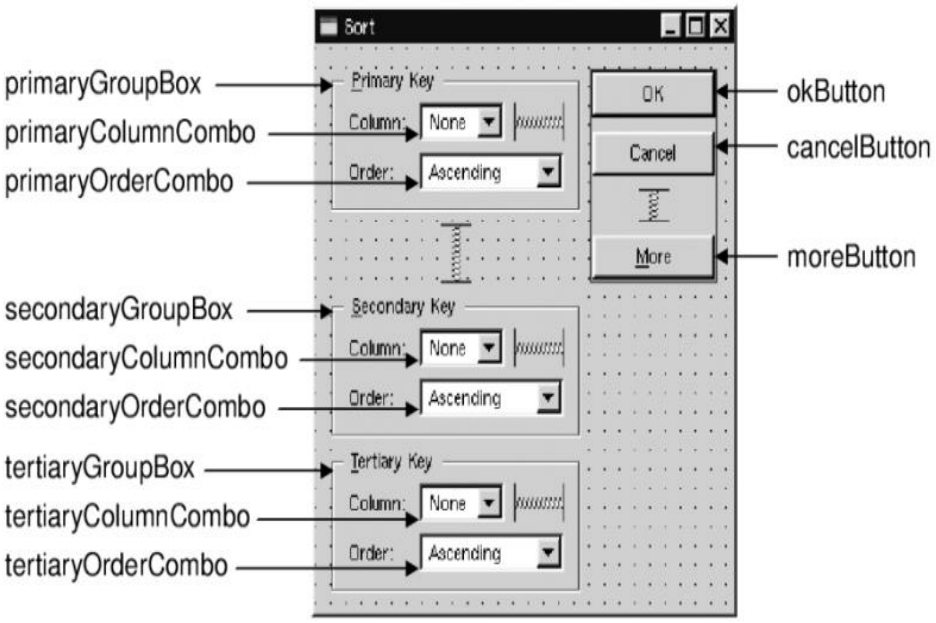
1. Растяните форму диалога, чтобы хватило места для дополнительных виджетов. Выберите GroupBox и скопируйте его в буфер обмена, выбрав пункт меню Edit|Copy. Вставьте новые группы виджетов, дважды выбрав пункт меню Edit|Paste. Переместите новые GroupBox-ы на свои места. Измените у них свойство **title**.
2. Создайте кнопки "OK", "Cancel" и "More".
3. Для кнопки "OK" установите свойство **default** в TRUE.
4. Добавьте две вертикальные распорки.
5. Расположите кнопки "OK", "Cancel" и "More" по вертикали. Переместите одну из распорок так, чтобы она встала между кнопками "Cancel" и "More". Выделите все четыре элемента и выберите пункт меню Layout|Lay Out Vertically.
6. Вторую распорку переместите так, чтобы она встала между первым и вторым GroupBox.
7. Установите свойство **sizeHint** у вертикальных распорок в значение (20, 10).
8. Разместите виджеты так, как это показано на рисунке.
9. Выберите пункт меню Layout|Lay Out in a Grid. У вас должна получиться заготовка, показанная на рисунке.



В результате такого размещения мы получили "сетку" из двух колонок и четырех строк -- всего восемь ячеек. Первый GroupBox, левая вертикальная распорка, второй и третий GroupBox занимают по одной ячейке. Кнопки "OK", "Cancel", "More" и правая вертикальная распорка занимают две ячейки. И вправом нижнем углу диалога у нас остаются две пустых ячейки. Если у вас не получилась такая компоновка виджетов -- отмените ее и повторите попытку.

Проверьте свойство формы **resizeMode**. Оно должно быть установлено как "Fixed", благодаря чему пользователь не сможет растягивать окно диалога. Тогда, всю ответственность за размер окна диалога возьмут на себя менеджеры размещения, изменяя его в случае, когда подчиненные виджеты показываются или скрываются. Это гарантирует показ окна диалога с оптимальными размерами.

Дайте форме имя "SortDialog" и установите свойство **caption** в "Sort". Дайте имена виджетам, в соответствии с рисунком.



В заключение соединим сигналы и слоты:

1. Соедините **okButton clicked()** с **SortDialog accept()**
2. Соедините **cancelButton clicked()** с **SortDialog reject()**
3. Соедините **moreButton toggled(bool)** с **secondaryGroupBox setShown(bool)**
4. Соедините **moreButton toggled(bool)** с **tertiaryGroupBox setShown(bool)**

Щелкните мышкой по форме дважды, чтобы запустить редактор и добавьте следующий текст:

```
1 void SortDialog::init()
2 {
3     secondaryGroupBox->hide();
4     tertiaryGroupBox->hide();
5     setColumnRange('A', 'Z');
6 }

7 void SortDialog::setColumnRange(QChar first, QChar last)
8 {
9     primaryColumnCombo->clear();
10    secondaryColumnCombo->clear();
11    tertiaryColumnCombo->clear();
```

```
12    secondaryColumnCombo->insertItem(tr("None"));
13    tertiaryColumnCombo->insertItem(tr("None"));

14    primaryColumnCombo->setMinimumSize(
15          secondaryColumnCombo->sizeHint());
16    QChar ch = first;
17    while (ch <= last) {
18       primaryColumnCombo->insertItem(ch);
19       secondaryColumnCombo->insertItem(ch);
20       tertiaryColumnCombo->insertItem(ch);
21       ch = ch.unicode() + 1;
22    }
23  }
```

Функция init делает второй и третий GroupBox невидимыми.

Слот setColumnRange() инициализирует содержимое выпадающих списков, в соответствии с именами выделенных колонок в электронной таблице. Мы вставили элемент "None", во второй и третий выпадающий списки, на тот случай, если пользователь пожелает выполнить сортировку только по одному столбцу. Не смотря на то, что мы не создавали это слот в **Qt Designer**, тем не менее он его обнаружит самостоятельно, а uic создаст соответствующее объявление в определении класса SortDialog.

В строках 14 и 15 можно наблюдать один хитрый трюк, связаный с размещением компонента. Функция QWidget::sizeHint() возвращает "идеальный" размер виджета, который пробует соблюсти система размещения. Дело в том, что виджетам с различным содержимым могут быть заданы различные размеры. Для выпадающих списков это означает, что второй и третий списки, содержащие слово "None", могут иметь больший размер, чем первый, в котором указано односимвольное имя столбца. Чтобы избежать такой несогласованности, мы задаем мнимальный размер, для первого выпадающего списка, равный "идеальному" размеру второго.

Ниже приводится текст функции main(), которая устанавливает диапазон выделенных столбцов от "C" до "F" и затем вызывает диалог:

```
#include <qapplication.h>
#include "sortdialog.h"
int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   SortDialog *dialog = new SortDialog;
   app.setMainWidget(dialog);
   dialog->setColumnRange('C', 'F');
   dialog->show();
   return app.exec();
}
```

На этом мы завершаем рассмотрение расширяемого диалога. Из примера видно, что разработка расширяемых диалогов ненамного сложнее, чем обычных. Все что нам потребовалось добавить -- это кнопка перехода из режима в режим, несколько дополнительных сигналов и слотов, и фиксированный размер формы.

Другой тип диалогов, изменяющих свое представление -- это многостраничные диалоги. Создание многостраничных диалогов проходит еще проще. Такого рода диалоги можно строить несколькими способами:

- В качестве основы можно использовать QTabWidget. Сверху он имеет набор вкладок, которые управляются встроенным QWidgetStack.

- Можно использовать связку QListBox и QWidgetStack, в которой текущий элемент QListBox-а определяет страницу в QWidgetStack.
- Или связку из классов QListView или QIconView и QWidgetStack, объединяемые так же как и в случае с QListBox.

Лабораторная работа № 7
Тема: графический интерфейс Qt.
Цель: разработка кода программного модуля, создающее простое окно для редактирования текста.

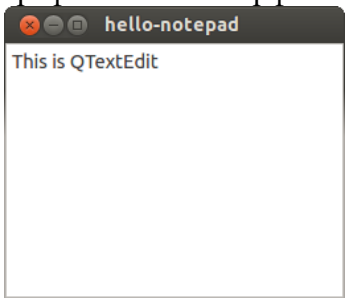
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Создадим простое окно для редактирования текста. Это самая элементарная программа с графическим интерфейсом Qt.



Исходный код примера:

```
#include <QApplication>
#include <QTextEdit>
```

```
Int main(int argv, char  **args)
{
QApplication app(argv, args);
QTextEdit  textEdit;
textEdit.show();
return app.exec();
}
```

Рассмотрим подробно каждую строчку кода. В первых двух, мы подключаем заголовочные файлы классов QApplication и QTextEdit, которые необходимы для работы этого примера. Каждому классу в Qt соответствует заголовочный файл с таким же названием.
В строке 6 создается объект класса QApplication, который управляет основными ресурсами, необходимыми для запуска любой программы с графическим интерфейсом Qt. Ему необходимо передать аргументы argv и args функции main(), т.к. Qt использует некоторые параметры командной строки, передаваемые при запуске программы.
В восьмой строке кода создается объект класса QTextEdit. QTextEdit — это визуальный элемент графического интерфейса. В Qt используются определенные виджеты — например, полосы прокрутки, метки и кнопки radio. Виджет может быть контейнером для хранения других виджетов. Наглядным примером является главное диалоговое окно программы.
В строке 9, окно редактирования текста выводится на экран в главном фрейме программы. Поскольку виджеты могут работать как контейнеры (В экземпляре класса QMainWindow находятся полосы панели инструментов, меню, строка состояния и несколько других виджетов), мы может отобразить только один виджет в окне нашей программы. По умолчанию, виджеты не видны. Функция show()используется для их отображения.
В строке 11, создается объект QApplication, который генерирует цикл событий в процессе работы приложения. События генерируются и передаются на обработку виджетам. Примерами событий могут являться клики мыши, нажатия клавиш на клавиатуре и т.п. Когда вы вводите текст в окне редактирования виджета QTextEdit, нажатия клавиш обрабатываются средствами виджета, и вводимый текст отображается в процессе набора.
Для запуска программы, откройте командную строку и зайдите в директорию с .cpp файлом программы. Выполните следующие команды shell-интерпретатора, чтобы скомпилировать пример.

```
Qmake –project
Qmake
Make
```

После успешного выполнения предыдущих команд, скомпилированная программа будет размещена в директории текущего проекта (в Windows вы можете использовать nmake вместо make. Исполняемые файлы будут размещены в директориях debug или release, которые создадутся командой make. qmake— это утилита, которая создает файлы конфигурации Qt-проекта, если ей передан аргумент-project. После создания файла конфигурации (.pro), qmake генерирует Makefile, который используется утилитой make для сборки приложения. Позже, мы рассмотрим процесс написания собственных .pro файлов.

Лабораторная работа № 8

Тема: Добавление кнопки выхода
Цель: разработка кода программного модуля, использующего виджет QTextEdit и кнопку входа QPushButton.

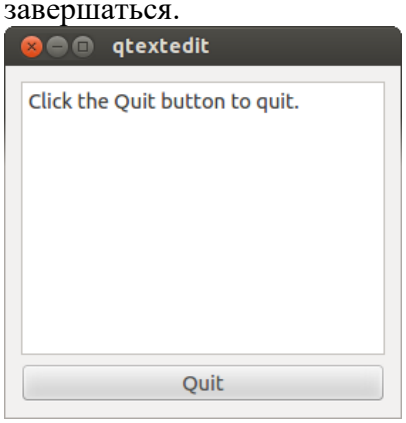
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

В реальных проектах, обычно используется более одного виджета. Добавим кнопку QPushButton под полем редактирования текста. При клике на нее, программа будет завершаться.



Рассмотрим исходный код программы.

```
#include <QtGui>
//#include <QTextEdit>
int main(int argc, char **argv)
{
   QApplication app(argc, argv);
   QTextEdit textEdit;
   QPushButton quitButton("Quit");
   QObject::connect(&quitButton, SIGNAL(clicked()), qApp,SLOT(quit()));
   QVBoxLayout layout;
   layout.addWidget(&textEdit);
   layout.addWidget(&quitButton);
QWidget window;
```

```
window.setLayout(&layout);
   window.show();

   return app.exec();
}
```

Сначала подключается заголовочный файл QtGui. В нем содержатся все классы графического интерфейса Qt.
В строке 10 применяется механизм сигналов и слотов, для того, чтобы закрыть программу после нажатия на кнопку выхода. Слот — это функция, которая может быть вызвана в процессе выполнения программы. Сигнал — функция, вызывающая функции-слоты, которые с ним посредствомQObject::connect.
quit() — слот класса QApplication, завершающий работу приложения. clicked() — сигнал, который передает нажатая кнопка QPushButton. Статическая функция QObject::connect связывает определенный слот и сигнал. SIGNAL() и SLOT() — макросы, которые принимают сигнатуры сигналов и слотов, для их связи между собой. Также, функции connect() необходимо передать указатели на объекты, которые будут принимать и рассылать сигналы.
В строке 12 создается объект класса QVBoxLayout. Как уже было сказано, виджеты могут содержать в себе другие виджеты. Можно задавать координаты и размер вложенных виджетов непосредственно, но обычно для этого используют слои (layouts). Слой управляет границами вложенных виджетов. Например, объект QVBoxLayout размещает виджеты в одну вертикальную колонку.
В строках 13 и 14, мы добавляем поле редактирования текста и кнопку выхода к слою layout.
В строке 17 задается главный слой для всего приложения.

Лабораторная работа № 9

Тема: Наследование QWidget

Цель: разработка кода программного модуля с наследником QWidget и добавление в него слота, который будет привязан к кнопке выхода.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного	

	программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Когда пользователь хочет закрыть приложение, ему можно показать всплывающее окно, в котором программа спросит подтверждение данного действия. Создадим наследник QWidget и добавим в него слот, который будет привязан к кнопке выхода.

Рассмотрим следующий код.

```
Class Notepad: public QWidget
{ Q_OBJECT
Public:
Notepad();
Private slots:
  Void quit();
Private:
  QTextEdit *textEdit;
  QPushButton *quitButton;
};
```

Макрос **Q_OBJECT** объявляет класс как QObject. Он должен находиться в самом начале определения класса. QObject добавляет несколько расширенных возможностей к обычному классу C++. Например, имена классов и слотов можно запросить в процессе выполнения. В строке 13 объявляется слот quit(). В последствии, мы присоединим этот слот к сигналу.
В предыдущих примерах, создание графики и присоединение слотов осуществлялось внутри функцииmain(). Сейчас мы будем использовать конструктор Notepad.

```
Notepad::Notepad()
{
 textEdit = new QTextEdit;
 quitButton = new QPushButton(tr(“Quit”));
 connect(quitButton,SIGNAL(clicked()),this,SLOT(quit()));
QVBoxLayout * layout = new QVBoxLayout;
 Layout->addWidget(textEdit);
 Layout->addWidget(quitButton);
 setLayout(layout);
 setWindowTitle(tr(“Notepad”));
}
```

Как видно из определения класса, мы используем указатели на объекты textEdit и quitButton. Для объектов QObject, почти всегда рациональнее выделять память в куче, вместо их копирования.
Мы используем функцию tr() для обработки всех строк, которые видны пользователю. Эта функция применяется, когда приложение нужно перевести на несколько языков. Здесь мы не

будем углубляться в детали, но вы можете найти нужную информацию в описании библиотеки Qt Linguist.

Создание файлов .pro

Ранее, мы использовали команду qmake -project для создания файлов .pro. В следующем примере, мы создадим этот файл вручную.

```
HEADERS=notepad.h
SOURCES=notepad.cpp\
main.cpp
```

Теперь для сборки программ на Qt, будут использоваться следующие команды.

```
qmake;
make;
```

Лабораторная работа № 10

Тема: Использование QMainWindow.

Цель: разработка кода программного модуля, с использованием виджета QMainWindow.

Время выполнения: 80 минут

Проверяемые результаты обучения

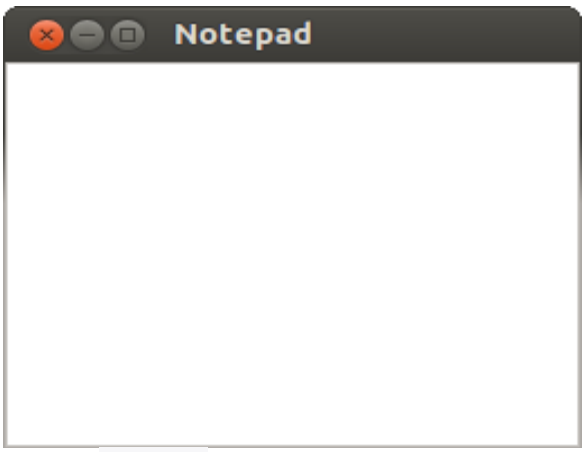
Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.

За не полностью выполненную работу выставляется – 1 балл.

За невыполненную работу выставляется – 0 баллов.

Во многих приложениях удобно использовать QMainWindow — слой, в который можно добавлять контекстные меню, встроенные виджеты, панели инструментов и строку состояния. В QMainWindowесть центральная область, куда можно добавлять любые дочерние виджеты. В этом примере мы разместим там поле для редактирования текста.



Рассмотрим определение класса Notepad.

```
#include <QtGui>

class Notepad : public QMainWindow
{
    Q_OBJECT;

public :
    Notepad();

private slots:
    void open();
    void save();
    void quit();

private :
    QTextEdit *textEdit;
    QAction *saveAction;
```

```
    QAction *exitAction;

    QMenu *fileMenu;

};
```

Мы создали два дополнительных слота — open() и save(). Они будут открывать и сохранять документ. Пример их реализации будет представлен позже.

Очень часто, один и тот же слот используется одновременно несколькими виджетами. Например, в пунктах меню и соответственных кнопках на панели инструментов. Чтобы упростить этот процесс, в Qt существует класс QAction, который можно передавать сразу нескольким виджетам и присоединять к слоту. Например, QMenu и QToolBar могут создавать пункты меню, используя одно действиеQAction. Скоро вы убедитесь, как это облегчает процессе разработки.

Сначала, мы используем конструктор класса Notepad для создания графического интерфейса программы.

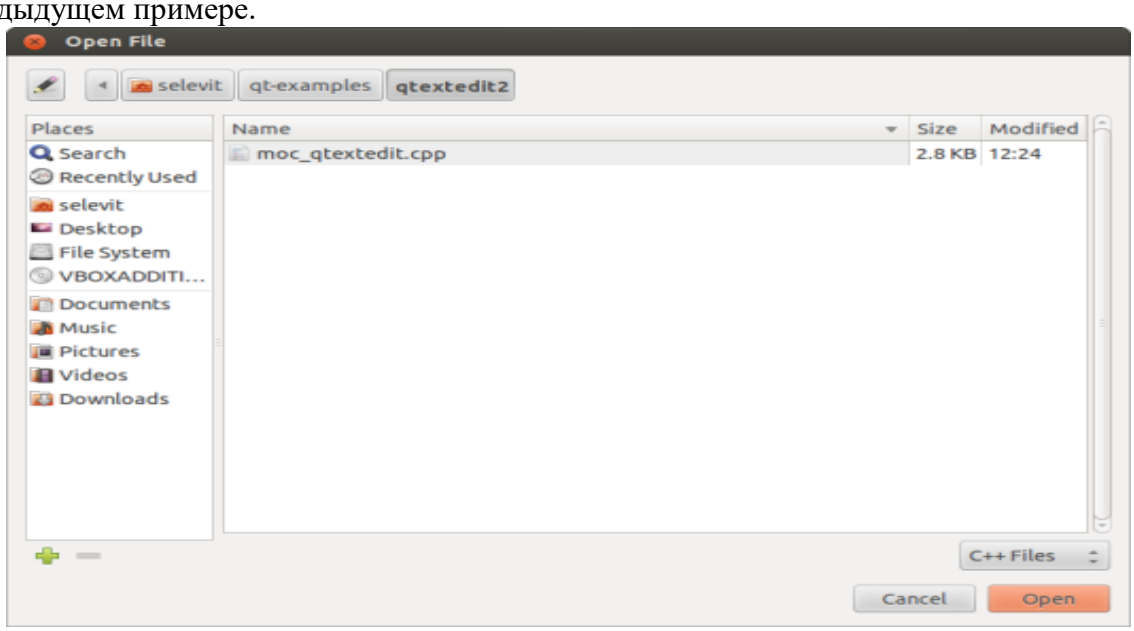
```
Notepad::Notepad()

{

    openAction = new QAction(tr( "&Open" ), this );

    saveAction = new QAction(tr( "&Save" ), this );

    exitAction = new QAction(tr( "E&xit" ), this );

    connect(openAction, SIGNAL(triggered()), this , SLOT(open()));

    connect(saveAction, SIGNAL(triggered()), this , SLOT(save()));

    connect(exitAction, SIGNAL(triggered()), qApp, SLOT(quit()));

    fileMenu = menuBar()->addMenu(tr( "&File" ));

    fileMenu->addAction(openAction);

    fileMenu->addAction(saveAction);
```

```
    fileMenu->addSeparator();
    fileMenu->addAction(exitAction);

    textEdit = new QTextEdit;
    setCentralWidget(textEdit);

    setWindowTitle(tr("Notepad"));
}
```

При создании действий, задается текст, отображаемый внутри виджетов, в которые будут добавлены эти действия (в нашем случае, пункты меню). Если нам нужно добавить кнопки для этих действий в панель инструментов, мы можем передать иконку для каждого действия. Когда пользователь кликает на пункте меню, рассылается сигнал действия и вызывается соответствующий слот.

Сохранение и открытие

В этом примере, мы реализуем функционал слотов open() и save(), которые были добавлены в предыдущем примере.




```
void Notepad::open()
{
    QString fileName = QFileDialog::getOpenFileName(this, tr("Open File"), "",
```

```
        tr(    "Text Files (*.txt);;C++ Files (*.cpp *.h)"    ));

    if    (fileName !=    ""    ) {
        QFile file(fileName);
        if    (!file.open(QIODevice::ReadOnly)) {
            QMessageBox::critical(    this    , tr(    "Error"    ), tr(    "Could not open file"    ));
            return    ;
        }
        QTextStream in(&file);
        textEdit->setText(in.readAll());
        file.close();
    }
}
```

Первый шаг — это запрос у пользователя имени файла для открытия. В состав Qt входит классQFileDialog — диалоговое окно, в котором пользователь может выбрать файл. Статическая функцияgetOpenFileName() открывает модальное окно для выбора файла и возвращает путь к выбранном файлу. Если пользователь отменил действие, функция возвращает пустую строку.

После того, как было получено имя файла, мы пробуем открыть его с помощью функции open(). Она возвращает логическое true в случае успешного завершения. Если файл открыть не удалось, мы используем класс QMessageBox для того, чтобы показать пользователю всплывающее окно с сообщением об ошибке (смотрите описание класса QMessageBox, для получения более подробной информации).

Чтение данных становится тривиальной задачей, благодаря классу QTextStream — оберткой над объектом QFile. Функция readAll() возвращает содержимое файла, как объект QString. Это содержимое мы поместим в поле для редактирования текста. Затем, мы закрываем файл с помощью функции close(), чтобы вернуть файловый дескриптор операционной системе.

Перейдем к реализации слота save().

```
void    Notepad::save()
```

```
{
    QString fileName = QFileDialog::getSaveFileName( this , tr( "Save File" ), "" ,
    tr( "Text Files (*.txt);;C++ Files (*.cpp *.h)" ));

    if (fileName != "" ) {
        QFile file(fileName);
        if (!file.open(QIODevice::WriteOnly)) {
            // error message
        } else {
            QTextStream stream(&file);
            stream << textEdit->toPlainText();
            stream.flush();
            file.close();
        }
    }
}
```

Для сохранения данных мы снова используем класс QTextStream. При помощи него, можно записывать строки QString в файл, используя оператор <<.

Лабораторная работа № 11

Тема: использование Qt layout для оптимизации интерфейса программы.

Цель: разместить компонент на **Qt layout**

Время выполнения: 80 минут

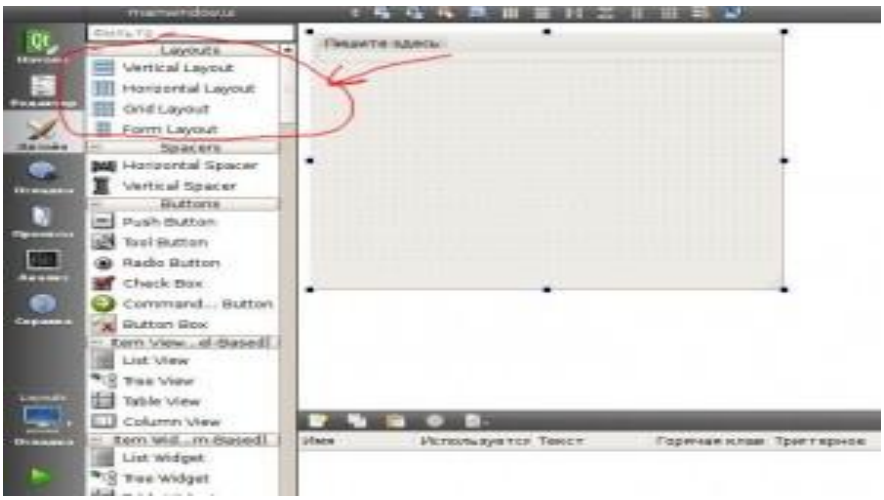
Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во

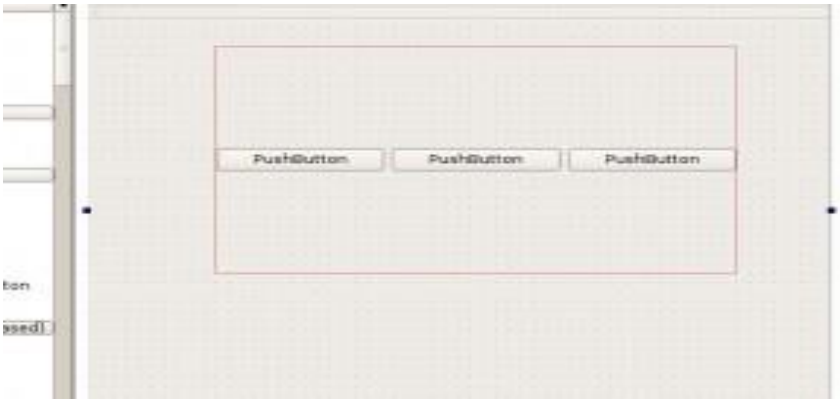
		баллов)
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

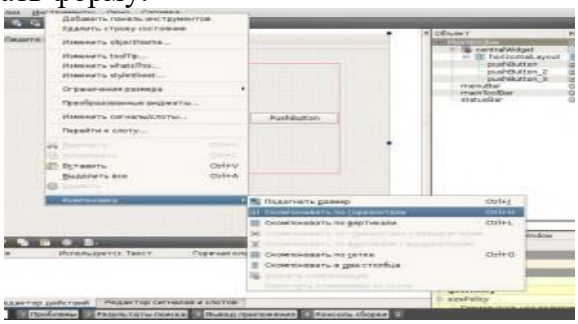
Средства **Qt** позволяют автоматически **оптимизировать интерфейс программы** для любого разрешения монитора или изменения размера окна. Сделать это можна с помощью**Qt layout**. При добавлении компонентов обычным методом (перетягиванием компонента в **дизайнере форм**) он имеет фиксированый размер. Это может вызвать нежелательные артефакты при изменении размеров окна. Если же разместить компонент на **Qt layout**, размер компонента будет автоматически подстраиваться под окно пользователя. **Qt layout** бывает нескольких видов, в зависимости от нужд пользователя и желания феншуйности интерфейса.



Чтобы начать использовать **Qt layout**, нужно в **Form Designer** добавить необходимый layout (см. рис. выше), потом разместить туда необходимые компоненты.



Можно разместить несколько **Qt layout** на форме в зависимости от нужд.
Дальше, для автоматического изменения размера **Qt layout** 'ов при изменении размера окна необходимо скомпонировать форму:



Можно неограничено вкладывать **Qt layout**'ы друг в друга. Если необходимо сохранить размер отдельно компонента, можно разместить его внутри отдельного **Qt Widget**, вложеного в **Qt layout**.

Лабораторная работа № 12
Тема: Создание приложения Qt на C++
Цель: написание программы Text Finder с использованием Qt Creator.
Время выполнения: 80 минут

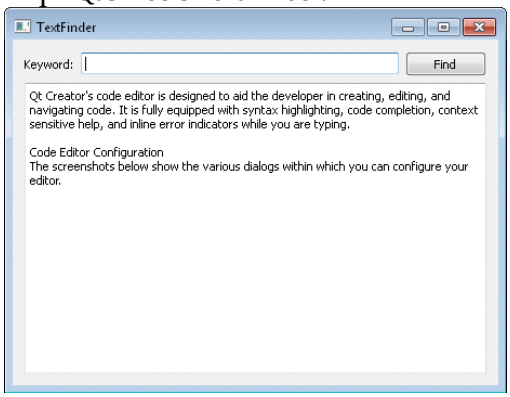
Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного	

	программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

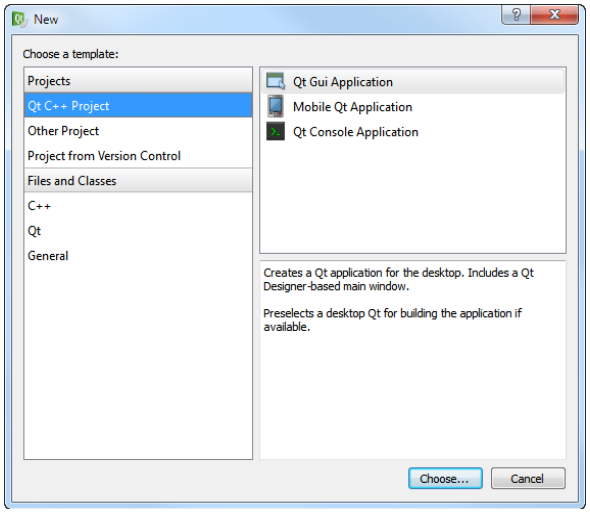
Это упрощённая версия примера QtUiTools Text Finder.



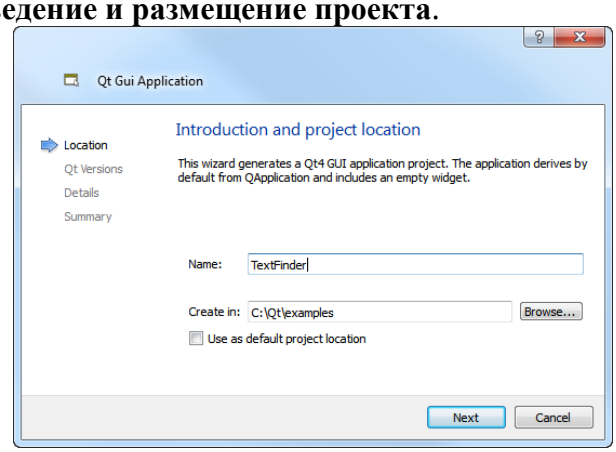
Создание проекта Text Finder

Замечание: Создайте проект с активным режимом **Справка** чтобы вы могли следовать этим инструкциям во время работы.

1. Выберите **Файл > Новый файл или проект... > Проект Qt C++ > GUI приложение Qt > Выбрать...**.

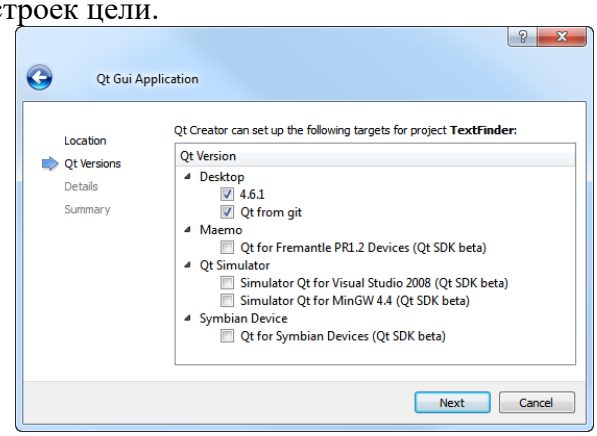


Откроется диалог **Введение и размещение проекта**.

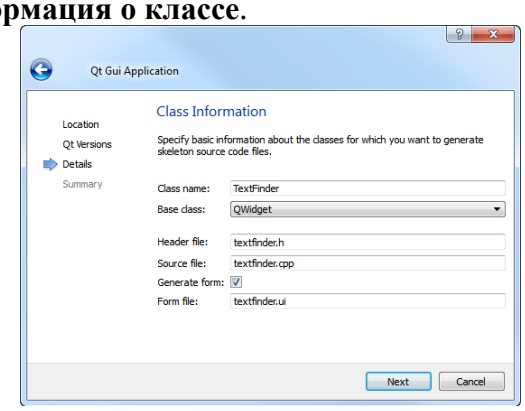


2. В поле **Имя** введите **TextFinder**.

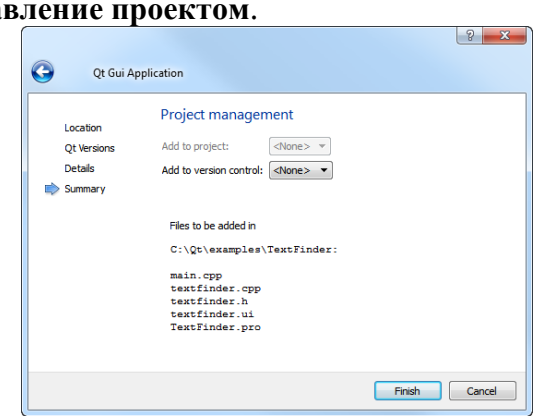
3. В поле **Создать в** введите путь к файлам проекта. Например, C:\Qt\examples, и нажмите **Вперёд**.
   Откроется диалог настроек цели.



4. Выберите нужный профиль Qt для сборки вашего проекта и нажмите **Далее**.
   Замечание: Если у вас установлен только один профиль Qt, этот диалог будет пропущен.
   Откроется диалог **Информация о классе**.



5. В поле **Имя класса** введите **TextFinder** в качестве имени класса.
6. В списке **Базовый класс** выберите **QWidget** в качестве типа базового класса.
   Замечание: Поля **Заголовочный файл**, **Файл исходников** и **Файл формы** будут обновлены автоматически в соответствии с выбранным вами именем класса.
7. Нажмите **Вперёд**.
   Откроется диалог **Управление проектом**.

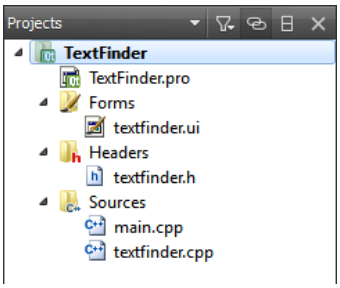


8. Проверьте настройки проекта и нажмите **Завершить** для создания проекта.

Проект TextFinder будет содержать следующие файлы:

- textfinder.h
- textfinder.cpp
- main.cpp

- textfinder.ui
- textfinder.pro

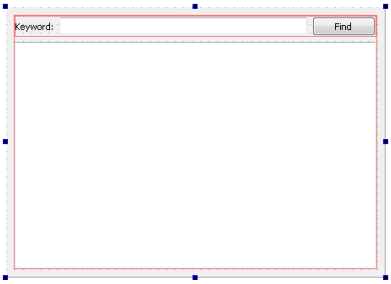


Файлы .h и .cpp содержат необходимые строки кода. Файл .pro полностью завершён.

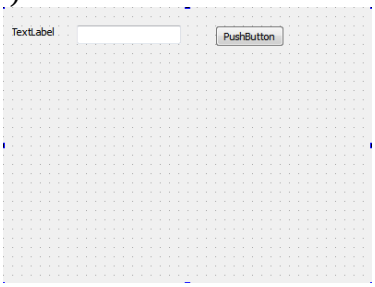
Заполнение недостающих кусков

Начнём с проектирования интерфейса пользователя и затем перейдём к заполнению недостающего кода. В заключение добавим поиск.

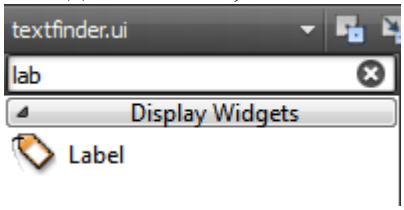
Проектирование пользовательского интерфейса



1. В режиме **Редактор** дважды нажмите на файле textfinder.ui в виде **Проекты** для запуска интегрированного *Qt Designer*.
2. Перетащите следующие виджеты на форму:
    1. **Label** (QLabel)
    2. **Line Edit** (QLineEdit)
    3. **Push Button** (QPushButton)



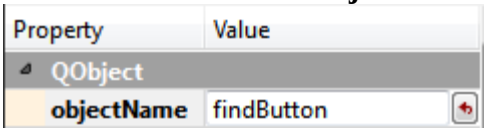
Замечание: Для быстрого поиска виджетов, используйте строку поиска в **боковой панели**. Например, для поиска виджета **Label**, начните печатать слово **label**.



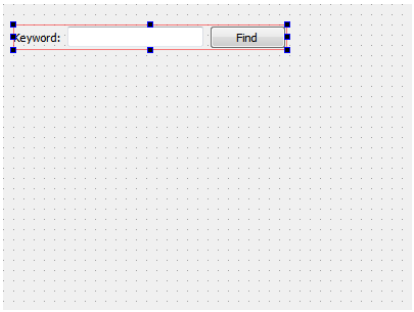
Дважды нажмите на виджет **Label** и введите текст **Keyword**.

Дважды нажмите на виджет **Push Button** и введите текст **Find**.

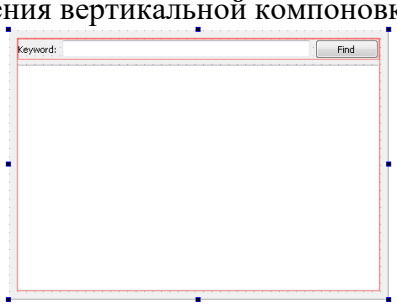
В панели **Свойства** измените значение свойства **objectName** на **findButton**.



Нажмите **Ctrl+A** для выбора виджетов и нажмите **Скомпоновать по горизонтали** (или нажмите **Ctrl+H**) для применения горизонтальной компоновки (QHBoxLayout).



Перетащите виджет **Text Edit** (QTextEdit) на форму.
Выберите область экрана и нажмите **Скомпоновать по вертикали** (или нажмите **Ctr+L**) для применения вертикальной компоновки (QVBoxLayout).



Применение вертикальной и горизонтальной компоновок обеспечивает масштабирование интерфейса приложения при различных разрешениях экрана.
Для вызова функции поиска при нажатии пользователем кнопки **Find** вы будете использовать механизм сигналов и слотов Qt. Сигнал вырабатывается когда происходит определённое событие, а слот - это функция которая вызывается в ответ на определённый сигнал. У виджетов Qt есть предопределённые сигналы и слоты которые вы можете использовать прямо из *Qt Designer*. Чтобы добавить слот для функции поиска:

0. Нажмите правой кнопкой мыши на кнопке **Find** для открытия контекстного меню.
1. Выберите **Перейти к слоту... > clicked()**, и нажмите **OK**.
   Закрытый слот on_findButton_clicked() будет добавлен в заголовочный файл textfinder.h и закрытая функция TextFinder::on_findButton_clicked() будет добавлена в файл исходных кодов textfinder.cpp.

Нажмите **Ctrl+S** для сохранения изменений.
Для получения дополнительной информации о проектировании форм с *Qt Designer* смотрите Руководство по Qt Designer.

Завершение заголовочного файла

Файл textfinder.h уже содержит необходимые директивы #include, конструктор, деструктор и объект Ui. Вам потребуется добавить закрытую функцию loadTextFile() для чтения и отображения содержимого входного файла в QTextEdit.
В боковой панели **Проекты** в режиме **Редактор** нажмите два раза на файле textfinder.h чтобы открыть его для редактирования.
Добавьте закрытую функцию в секцию private за Ui::TextFinder, как продемонстрировано в следующем фрагменте кода:

```
private slots:
    void on_findButton_clicked();
```

```
private:
    Ui::TextFinder *ui;
    void loadTextFile();
```

Завершение файла исходных кодов

Теперь заголовочный файл завершён, перейдём к файлу исходных кодов textfinder.cpp.

1. В боковой панели **Проекты** в режиме **Редактор** нажмите два раза на файле textfinder.cpp чтобы открыть его для редактирования.
2. Добавьте код для загрузки текстового файла с помощью QFile, чтения его с помощью QTextStream, а затем отображения его в textEdit с помощью setPlainText(). Это продемонстрировано в следующем фрагменте кода:

```
void TextFinder::loadTextFile()
{
    QFile inputFile(":/input.txt");
    inputFile.open(QIODevice::ReadOnly);

    QTextStream in(&inputFile);
    QString line = in.readAll();
    inputFile.close();

    ui->textEdit->setPlainText(line);
    QTextCursor cursor = ui->textEdit->textCursor();
    cursor.movePosition(QTextCursor::Start, QTextCursor::MoveAnchor, 1);
}
```

3. Для использования QFile и QTextStream добавьте следующие директивы #includes в textfinder.cpp:

```
#include <QtCore/QFile>
#include <QtCore/QTextStream>
```

4. В слоте on_findButton_clicked() добавьте код извлечения строки поиска и использования функции find() для поиска строки в текстовом файле. Это продемонстрировано в следующем фрагменте кода:

```
void TextFinder::on_findButton_clicked()
{
    QString searchString = ui->lineEdit->text();
    ui->textEdit->find(searchString, QTextDocument::FindWholeWords);
}
```

5. После того как эти функции завершены, добавьте строку для вызова loadTextFile() в конструкторе, как продемонстрировано в следующем фрагменте кода:

```
TextFinder::TextFinder(QWidget *parent)
    : QWidget(parent), ui(new Ui::TextFinder)
{
    ui->setupUi(this);
    loadTextFile();
}
```

Слот on_findButton_clicked() будет вызван автоматически в сгенерированном uic ui_textfinder.h этой строчкой кода:

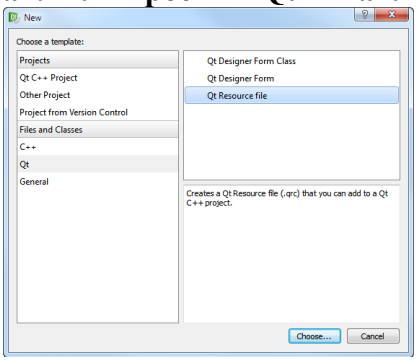
```
QMetaObject::connectSlotsByName(TextFinder);
```

Создание файла ресурсов

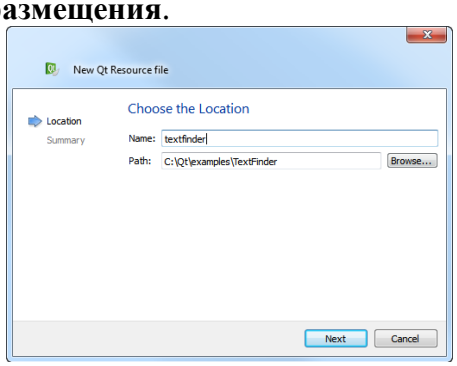
Вам потребуется файл ресурсов (.qrc) в котором вы сохраните входной текстовый файл. Входным файлом может быть любой файл .txt с абзацем текста. Создайте текстовый файл с именем input.txt и сохраните его в каталоге textfinder.

Чтобы добавить файл ресурсов:

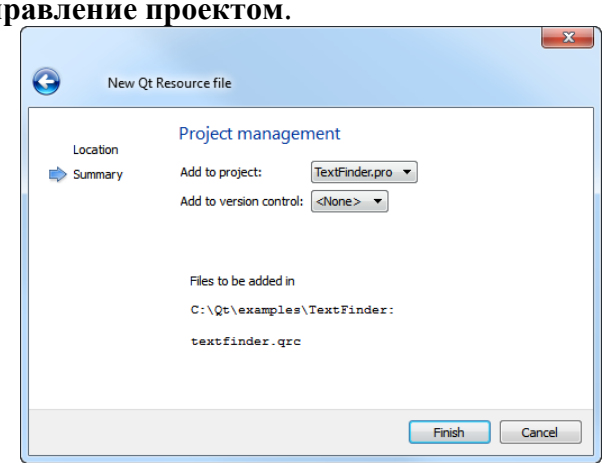
1. Выберите **Файл > Новый файл или проект > Qt > Файл ресурсов Qt > Выбрать...**.



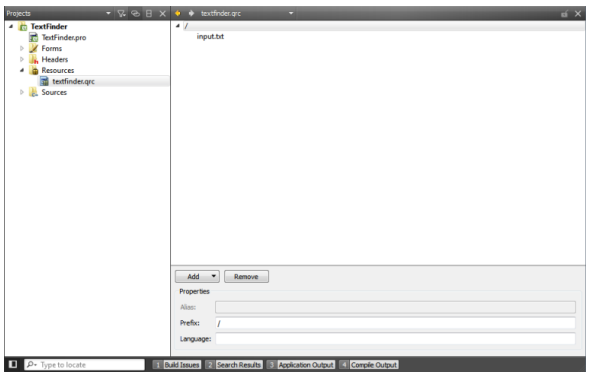
Откроется диалог **Выбор размещения**.



2. В поле **Имя** введите **textfinder**.
3. В поле **Путь** введите C:\Qt\examples\TextFinder и нажмите **Далее**.
   Откроется диалог **Управление проектом**.



4. В поле **Добавить в проект** выберите **TextFinder.pro** и нажмите **Завершить** для открытия файла в редакторе кода.
5. Выберите **Добавить > Добавить префикс**.
6. В поле **Префикс** замените префикс по умолчанию на (/).
7. Выберите **Добавить > Добавить файлы** для поиска и добавления input.txt.



Сборка и запуск вашего приложения

Теперь, когда у вас есть все необходимые файлы, нажмите на кнопку ▶ чтобы скомпилировать и запустить вашу программу.

Лабораторная работа №13

Тема: Механизм сигналов и слотов, позволяющий объектам взаимодействовать друг с другом.

Цель: Разработка программы в Qt Creator с графическим интерфейсом, использующей сигналы и слоты.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Важные классы Qt, не предназначенные для работы с графическим интерфейсом; надеюсь, что вы самостоятельно разобрались с классами, которые не обсуждались, поскольку секрет успешного изучения любой технологии заключается в самостоятельной работе. В этой статье

мы поработаем с механизмом, лежащим в основе поддержки графических интерфейсов в Qt - сигналами и слотами.

Механизм сигналов и слотов позволяет объектам взаимодействовать друг с другом. В повседневной жизни мы можем найти множество примеров подобного механизма; например, светофор - когда загорается красный свет, вы останавливаете свое транспортное средство. Красный свет является сигналом, а остановка транспортного средства - слотом. Когда светофор генерирует сигнал включения красного света, вы вызываете слот, т.е. вы останавливаете транспортное средство.

Для правильного понимания этого механизма нам необходимо попрактиковаться в работе с Qt - разработать нашу первую программу с графическим интерфейсом, использующую сигналы и слоты. Откройте Qt Creator. Перейдите в меню Файл -> Новый файл или проект -> Проект Qt Widget -> GUI приложение Qt (File -> New File or Project -> Qt Widget Project -> Qt GUI Application) (Рисунок 1). Выберите директорию для размещения и название вашего проекта (Рисунок 2). В качестве цели выберите Desktop (Рисунок 3). (картинки кликабельны)

Рис.1: Qt Creator

Рис.2: Выбор директории размещения

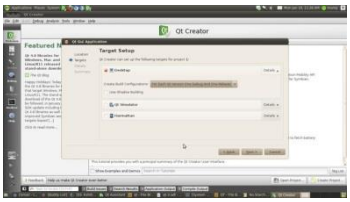


Рис.3: Выбор цели проекта

После этого нужно будет ответить на вопросы о классах. В поле выбора базового класса вы можете обнаружить три варианта, используемые для разных целей. Мы рассмотрим их в следующей статье. На данный момент следует выбрать QDialog в качестве базового класса (Рисунок 4). Дайте имя вашему классу, после чего нажмите "Далее", и, наконец, нажмите "Завершить".

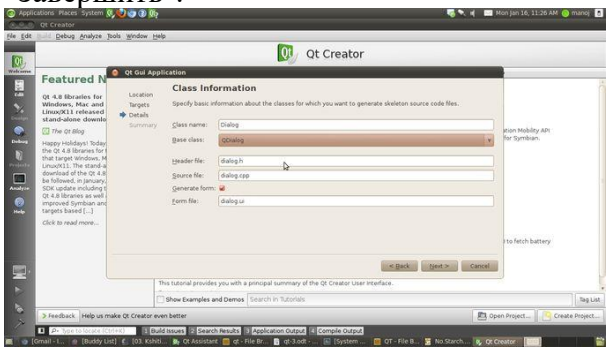


Рисунок 4: Создание класса

Сразу после этого Qt Creator откроет ваш новый проект (Рисунок 5) в окне проекта слева. В составе проекта должен присутствовать один файл проекта с названиемproject-name.pro, а в нашем случае 1.pro.

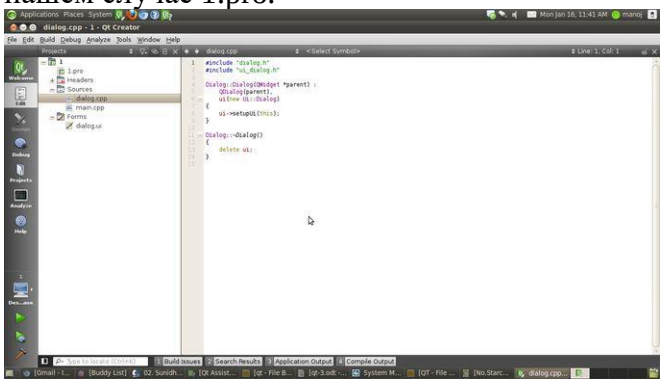


Рисунок 5: Редактор кода

Ниже расположены три директории: Заголовочные (Headers), Исходники (Sources) и Формы (Forms), в в которых находятся файлы соответствующих типов. Откройте любой файл при помощи двойного клика. Класс, которому мы дали название в процессе создания проекта, состоит из трех файлов: dialog.h (заголовочный файл), dialog.cpp(исходный код C++) и dialog.ui (файл пользовательского интерфейса или формы). Откройте файл пользовательского интерфейса в редакторе с помощью двойного клика (Рисунок 6).

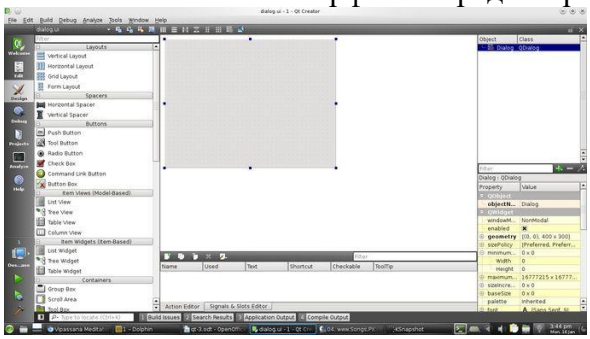
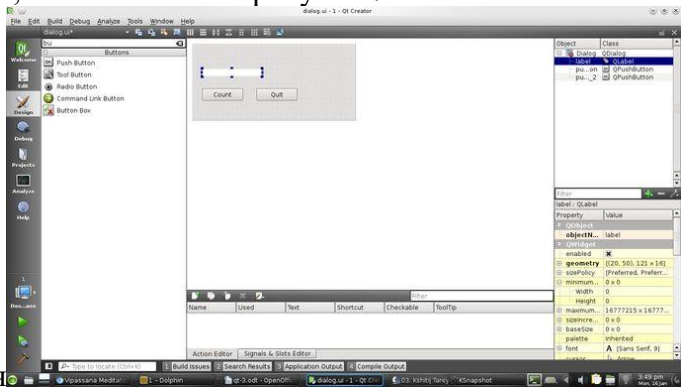


Рисунок 6: Редактор пользовательского интерфейса

Панель слева содержит список элементов графического интерфейса, которые могут быть помещены в окне диалога при помощи простого перетаскивания. Используем в нашем первом примере строку (Label) и две кнопки (Push Button). Окно диалога должно выглядеть так, как показано на рисунке 7.



Ия

Рисунок 7: Окно диалога

Для простоты не будем использовать элементы размещения (Layouts). При помощи двойных кликов измените текст кнопок на "Count" и "Quit"; используйте двойной клик по отношению к строке и удалите текст. Нажатие на кнопку "Count" должно увеличивать значение счетчика и выводить его в качестве текста строки, а нажатие на кнопку "Quit" - завершать приложение.

Теперь мы добавим слот для сигнала "clicked" кнопки "Quit". Нажмите F4 для перехода в режим редактора сигналов и слотов. Нажмите кнопку "Quit" и отпустите кнопку мыши на свободном пространстве окна диалога. После того, как вы отпустите кнопку мыши, должно появиться окно "Настройка соединения" (Configure Connection) (Рисунок 8). Слева перечислены сигналы, которые может генерировать элемент графического интерфейса; справа перечислены слоты (проще говоря, действия), которые могут быть выполнены с элементом графического интерфейса QDialog.

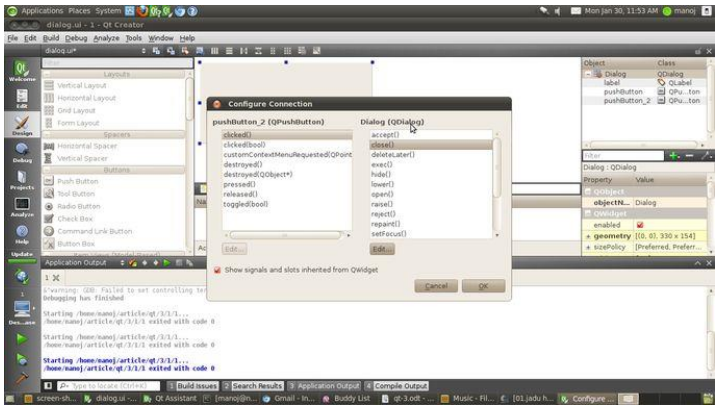


Рисунок 8: Настройка соединения

В левом списке выберите clicked(), а в правом - close(). Таким образом, при нажатии кнопки "Quit" окно будет закрыто. Нажмите F3 для перехода в режим редактирования свойств элементов графического интерфейса. Кнопка "Count" тоже должна быть соединена со слотом. Выберите эту кнопку, нажмите правой кнопкой мыши и во всплывающем меню выберите пункт "Перейти к слоту..." ("Go to Slot"). В появившемся окне (Рисунок 9) "Переход к слоту" ("Go to Slot") будут перечислены сигналы, генерируемые данным элементом графического интерфейса.

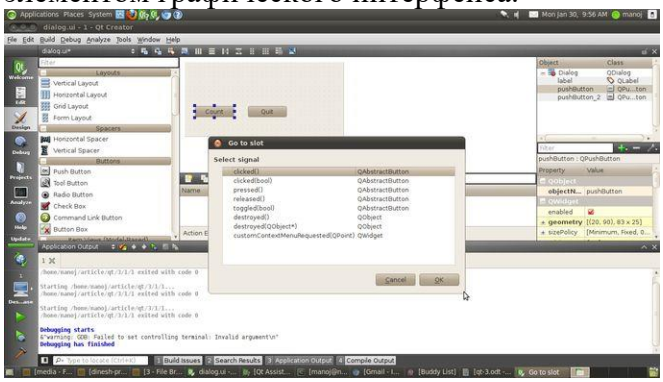


Рисунок 9: Переход к слоту

Выберите сигнал clicked() и нажмите OK. В редакторе кода будет открыта функция void Dialog::on_pushButton_clicked() из файла dialog.cpp (Рисунок 10).

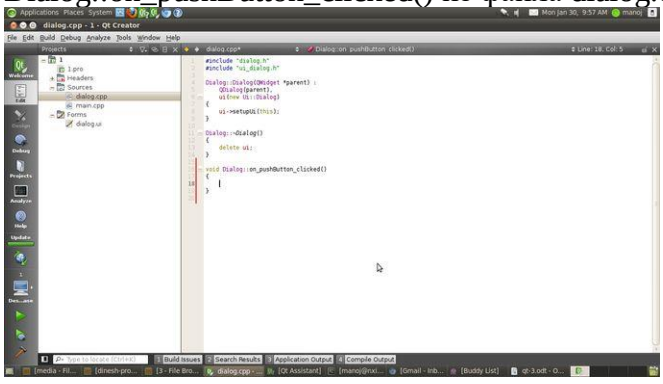


Рисунок 10: Функция слота

Отредактируйте этот код следующим образом:

```
void Dialog::on_pushButton_clicked()
{
        static int count;
        count++;
        ui->label->setText(QString::number(count));
}
```

В этом коде используется статическая переменная, значение которой увеличивается на 1 при каждом вызове функции, после чего значение преобразуется в строку QString и устанавливается в качестве текста строки. Нажмите Ctrl+R для сборки и запуска приложения (как на Рисунке 11).

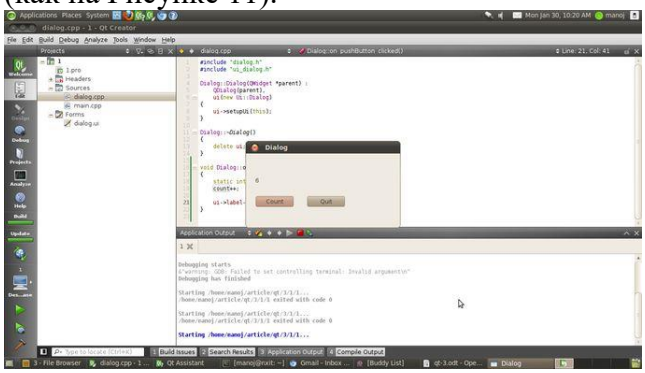


Рисунок 11: Вывод

Протестируйте приложение - нажмите кнопку "Count" и проверьте результат. Нажмите кнопку "Quit" для завершения работы приложения.

Лабораторная работа № 14

Тема: Элементы управления

Цель: Разработать приложение с элементами управления QPushButton и QLineEdit

Время выполнения: 80 минут

Проверяемые результаты обучения

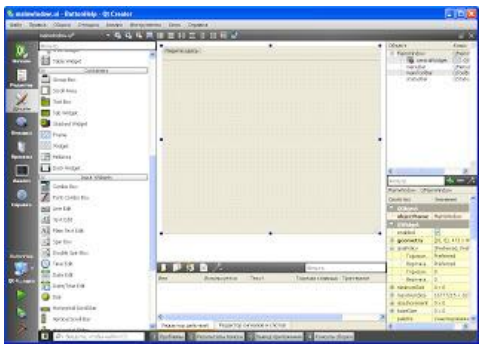
Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.

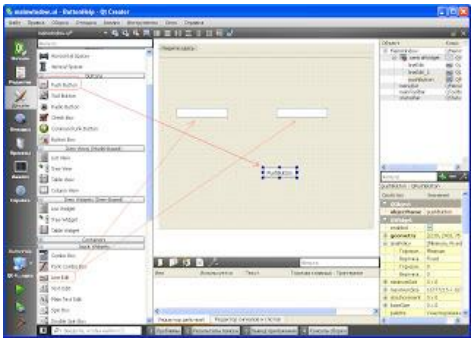
За не полностью выполненную работу выставляется – 1 балл.

За невыполненную работу выставляется – 0 баллов.

Открываем файл mainwindow.ui двойным кликом и нашим глазам предстанет следующая картина:



Следующим шагом добавим два QLineEdit и один QPushButton, т.е. 2 поля для ввода текста и одно кнопку. Будем имя кнопки завать слиянием текстовых значений в текстовых полях.



Для добавленных объектов существуют имена по умолчанию (поле objectName справа во второй половине списка свойств), в нашем случае это lineEdit и lineEdit_2 для текстовых полей и pushButton для кнопки.
В автоматически сгенерированных файлах к классу MainWindow уже есть указатель на форму:
Ui::MainWindow *ui;

В конструкторе уже прописана его инициализация:
```
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this); // инициализация указателя на форму
}
```

В соответствии с особенностью Qt в виде сигналов и слотов (http://easy-qt.blogspot.com/2012/01/qt_31.html) наша кнопка будет "сигналить" при клике по ней (да поможет вам справка). Для этого создадим слот, к которому в последующем и подключим наш сигнал. mainwindow.h будет выглядеть следующим образом:

```
#ifndef MAINWINDOW_H

#define MAINWINDOW_H

#include <QMainWindow>

#include <QString> // добавим эту библиотеку для работы со строками
```

```
namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

   Q_OBJECT

public:

   explicit MainWindow(QWidget *parent = 0);

   ~MainWindow();

private slots:

   void nashSlot(); // это созданный нами слот

private:

   Ui::MainWindow *ui;

};

#endif // MAINWINDOW_H
```

Теперь делаем определение этого слота в mainwindow.cpp

```
void MainWindow::nashSlot()
{
   QString str1 = ui->lineEdit->text(); // получаем строку из первого QLineEdit
   QString str2 = ui->lineEdit_2->text(); // получаем строку из второго QLineEdit
   QString finalStr = str1 + str2; // объединяем эти строки в одну
   ui->pushButton->setText(finalStr); // задаем имя кнопки равным финальной строке
}
```

После этого у вас ещё ничего не заработает, не торопитесь. Остался последний штрих - подключить сигнал от кнопки к нашему слоту. Сделаем это в конструкторе следующим образом:

```
MainWindow::MainWindow(QWidget *parent) :
   QMainWindow(parent),
   ui(new Ui::MainWindow)
{
   ui->setupUi(this);
   connect(ui->pushButton, SIGNAL(clicked()), this, SLOT(nashSlot()));
}
```

Это общее представление о том, как работать с кнопками и текстовыми полями. Если вам надо получить числовое значение из QLineEdit (а это бывает очень часто), делается это так:

```
int vall = ui->lineEdit->text().toInt();
```

А обратно

```
QString vall2 = QString::number(vall);
```

Лабораторная работа № 15

Тема: Окно с текстом

Цель: Разработать приложение с текстом в окне и сделать описание каждого оператора

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Ниже приводится текст оконного приложения Qt программы:

```
#include <QtCore/QCoreApplication>
#include <QDebug>

int main(int argc, char *argv[])
{
   QCoreApplication a(argc, argv);
```

```
    QString mStr = "Hello World";

    qDebug () << mStr;

    return a.exec();
}
```

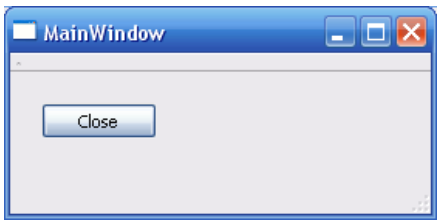
Ответить на вопросы:
Какие определения классов подключаются в данном приложении?
Что управляет ресурсами приложения?
Какие аргументы имеет конструктор QCoreApplication?
Какой визуальный компонент создаётся в приложении для отображения надписи "Hello World"?
Каким оператором выполняется передача управления библиотеке Qt. С этого момента программа переходит в режим ожидания, когда она ничего не делает, а просто ждет действий пользователя, например, нажатие на клавишу или кнопку мыши.
Зайти в папку 02 Hello World с исполняемым файлом данного приложения hello.exe и запустить его для выполнения.

Лабораторная работа № 16
Тема: Окно с текстом на элементе управления «кнопка»
Цель: Разработать приложение с текстом в окне и сделать описание каждого оператора
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.



Ниже приводится текст оконного приложения Qt программы:

Main.cpp

```
#include <QtGui/QApplication>
#include "mainwindow.h"
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

mainwindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->pushButton->setText("Close");
}

MainWindow::~MainWindow()
{
    delete ui;
}
```

Ответить на вопросы:

Какие определения классов подключаются в данном приложении?

Что управляет ресурсами приложения?

Какие аргументы имеет конструктор QApplication?

Какой визуальный компонент создаётся в приложении?

Каким оператором выполняется передача управления библиотеке Qt. С этого момента программа переходит в режим ожидания, когда она ничего не делает, а просто ждет действий пользователя, например, нажатие на клавишу или кнопку мыши.

Зайти в папку 03 Intro to GUI с исполняемым файлом данного приложения gui.exe и запустить его для выполнения.

Лабораторная работа № 17

Тема: Окно с элементами управления «кнопка» и «horizontalSlider»

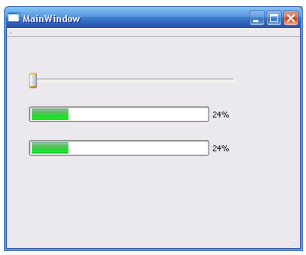
Цель: Разработать приложение с элементами управления в окне и сделать описание каждого оператора.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.



Ниже приводится текст оконного приложения Qt программы:

Main.cpp

```
#include <QtGui/QApplication>
#include "mainwindow.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
```

```
    MainWindow w;
    w.show();

    return a.exec();
}
```

mainwindow.cpp:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    connect(ui->horizontalSlider,SIGNAL(valueChanged(int)),
            ui->progressBar,SLOT(setValue(int)));

    connect(ui->horizontalSlider,SIGNAL(valueChanged(int)),
            ui->progressBar_2,SLOT(setValue(int)));
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pushButton_clicked(bool checked)
{

}

void MainWindow::on_pushButton_released()
{

}

void MainWindow::on_pushButton_toggled(bool checked)
{

}

void MainWindow::on_pushButton_pressed()
{
```

```
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

namespace Ui {
   class MainWindow;
}

class MainWindow : public QMainWindow
{
   Q_OBJECT

public:
   explicit MainWindow(QWidget *parent = 0);
   ~MainWindow();

private:
   Ui::MainWindow *ui;

private slots:
   void on_pushButton_pressed();
   void on_pushButton_toggled(bool checked);
   void on_pushButton_released();
   void on_pushButton_clicked(bool checked);
};

#endif // MAINWINDOW_H
```

Ответить на вопросы:
Какие определения классов подключаются в данном приложении?
Что управляет ресурсами приложения?
Какие аргументы имеет конструктор QApplication?
Какие визуальные компоненты создаются в приложении?
Какие сигналы и слоты используются?
Какие события обрабатываются?
Каким оператором выполняется передача управления библиотеке Qt. С этого момента программа переходит в режим ожидания, когда она ничего не делает, а просто ждет действий пользователя, например, нажатие на клавишу или кнопку мыши.
Задание:
Зайти в папку 04 Signals and Slots -> myprogress-build-desktop ->debug с исполняемым файлом данного приложения myprogress.exe и запустить его для выполнения.

Зайти в папку 04 Signals and Slots -> myprogress и запустить mainwindow.ui. Описать свойства каждого компонента.

Лабораторная работа № 18

Тема: Графические возможности Qt

Цель: Разработать приложение с изображением на "графических устройствах"
Двухмерная графика.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	
З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Краеугольным камнем движка двухмерной графики в Qt является QPainter. Он может использоваться для рисования на поверхности виджета (на экране), во внутреннем буфере (pixmap) и на принтере. Кроме того, в состав Qt входит класс QCanvas, который позволяет создавать изображения из графических примитивов.

В качестве альтернативы QPainter и QCanvas, можно рассматривать библиотеку OpenGL. Она предоставляет механизмы создания трехмерной графики, но может использоваться и для рисования двухмерных изображений. Код, использующий OpenGL очень легко интегрируется в приложения Qt.

Рисование средствами QPainter

Класс QPainter используется для создания изображений на "графических устройствах", таких как виджеты или карты пикселей (pixmap). Чаще всего он используется при создании нестандартных виджетов, для придания им уникального, ни на что не похожего, внешнего

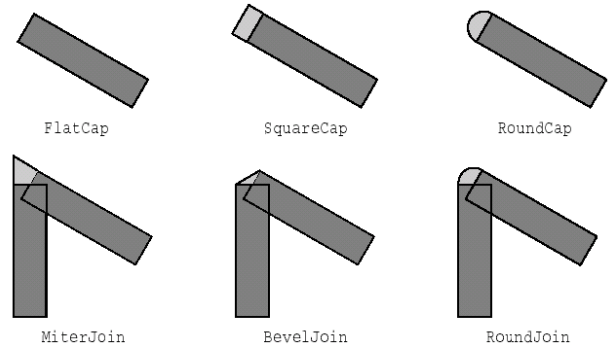
вида. Однако этот класс может использоваться и для вывода графики на принтер, более подробно мы коснемся этого вопроса немного ниже.

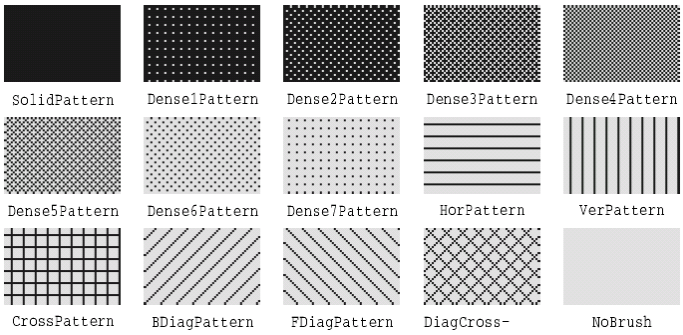
QPainter может рисовать простые геометрические фигуры: точки, линии, прямоугольники, эллипсы, дуги, сегменты круга, замкнутые ломаные (многоугольники) и кривые Безье. Он так же может отображать карты пикселей, рисунки и текст.

Тремя наиболее важными характеристиками QPainter являются **перо** (pen), **кисть** (brush) и **шрифт** (font).

- *Перо* используется для рисования линий и границ геометрических фигур. Оно характеризуется такими параметрами, как: цвет, толщина, стиль рисования линий, стиль оформления концов линий и стиль оформления углов.
- *Кисть* -- это шаблон, которым заполняются геометрические фигуры. Кисти характеризуются цветом и стилем.
- *Шрифт* используется для рисования текста. Шрифт может иметь огромное количество атрибутов, среди них: название и размер.

Настройки этих характеристик могут быть выполнены с помощью функций:
setPen(), setBrush() и setFont().





1. Итак, для того чтобы начать рисование, необходимо создать проект GUI приложение в Qt Creator, в качестве базового класса выбрать QDialog (вообще можно выбрать любой класс).

2. Войти в режим редактирования
3. Открыть заголовочный файл dialog.h и отредактировать код:

```
1. #ifndef DIALOG_H
2. #define DIALOG_H
3. #include <QDialog>
```

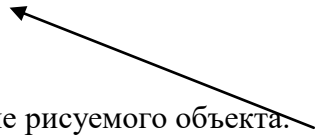
```
#include <QtGui>
#include <QtCore>
namespace Ui {
    class Dialog;
}
class Dialog : public QDialog
{
    Q_OBJECT
public:
    explicit Dialog(QWidget *parent = 0);
    ~Dialog();
private:
    Ui::Dialog *ui;
protected:
    void paintEvent(QPaintEvent *e);
};
#endif // DIALOG_H
```

3. Сохраните проект и откройте файл исходных кодов dialog.cpp.
После кода:

```
Dialog::~Dialog()
{
   delete ui;
}
```

Допишите код:

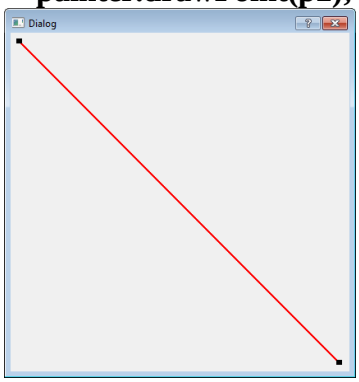
```
void Dialog::paintEvent(QPaintEvent *e)
{
//описание рисуемого объекта
}
```



Теперь можно начать описание рисуемого объекта.
Примеры описания рисуемых объектов**:**
Объект « Линия»

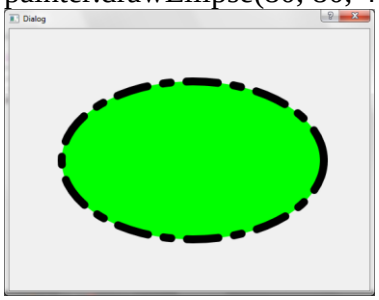
```
QPainter painter(this);

  QPen pointpen(Qt::black);
   pointpen.setWidth(6);

   QPen linepen(Qt::red);
   linepen.setWidth(2);

   QPoint p1;
   p1.setX(10);
   p1.setY(10);


   QPoint p2;
   p2.setX(100);
   p2.setY(100);
```

```
painter.setPen(linepen);
painter.drawLine(p1,p2);

painter.setPen(pointpen);
painter.drawPoint(p1);
painter.drawPoint(p2);
```



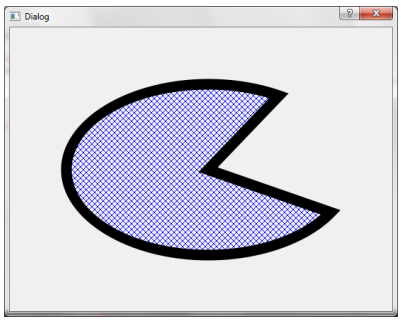
Ниже приводится код, который рисует объект «эллипс».

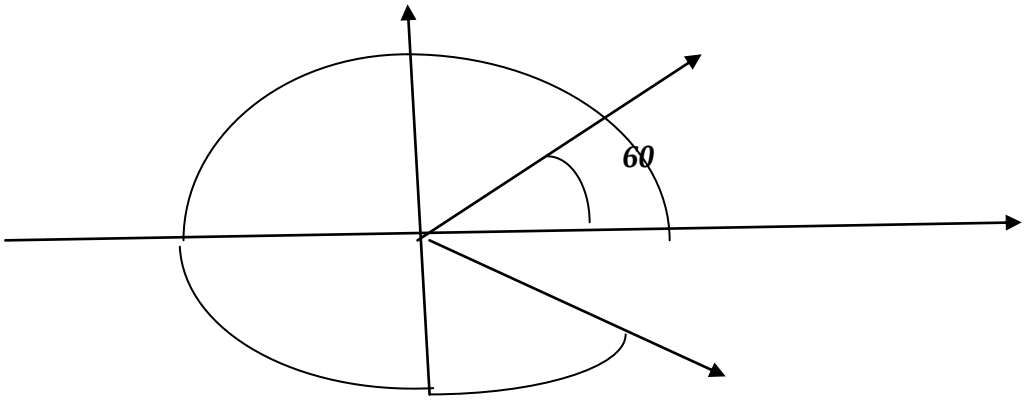
```
QPainter painter(this);
painter.setRenderHint(QPainter::Antialiasing, true);
painter.setPen(QPen(Qt::black, 12, Qt::DashDotLine, Qt::RoundCap));
painter.setBrush(QBrush(Qt::green, Qt::SolidPattern));
painter.drawEllipse(80, 80, 400, 240);
```



Следующий код рисует объект «сегмент круга»

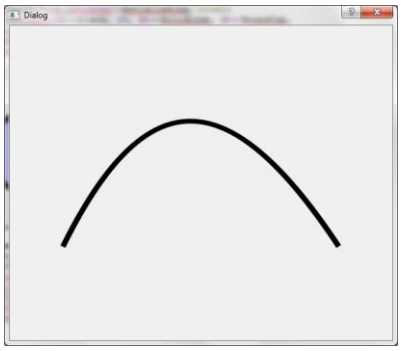
```
QPainter painter(this);
painter.setRenderHint(QPainter::Antialiasing, true);
painter.setPen(QPen(Qt::black, 15, Qt::SolidLine, Qt::RoundCap, Qt::MiterJoin));
painter.setBrush(QBrush(Qt::blue, Qt::DiagCrossPattern));
painter.drawPie(80, 80, 400, 240, 60 * 16, 270 * 16);
```





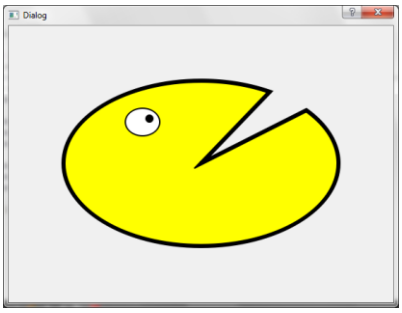
Код, который рисует кривую Безье

```
QPainter painter(this);
painter.setRenderHint(QPainter::Antialiasing, true);
QPainterPath path;
path.moveTo(80, 320);
path.cubicTo(200, 80, 320, 80, 480, 320);
painter.setPen(QPen(Qt::black, 8));
painter.drawPath(path);
```



Ещё пример

```
QPainter painter(this);
painter.setRenderHint(QPainter::Antialiasing, true);
painter.setPen(QPen(Qt::black, 6, Qt::SolidLine, Qt::RoundCap, Qt::MiterJoin));
painter.setBrush(QBrush(Qt::yellow));
painter.drawPie(80, 80, 400, 240, 60 * 16, 340 * 16);
painter.setPen(QPen(Qt::black, 2, Qt::SolidLine, Qt::RoundCap));
painter.setBrush(QBrush(Qt::white, Qt::SolidPattern));
painter.drawEllipse(170, 120, 50, 40);
painter.setPen(QPen(Qt::black, 2, Qt::SolidLine, Qt::RoundCap));
painter.setBrush(QBrush(Qt::black, Qt::SolidPattern));
```



painter.drawEllipse(200, 130, 10, 10)

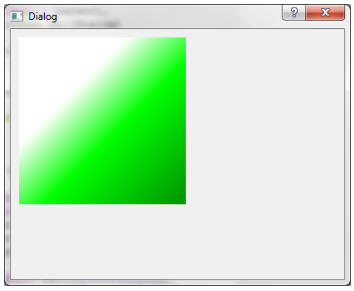
Градиенты

Для того чтобы нарисовать градиент допишите в заголовочном файле dialog.h следующие строки:

```
#include <QRadialGradient>
#include<QLinearGradient>
#include <QConicalGradient>
```

В файле исходного кода dialog.cpp в описании paintEvent вставить код описания рисуемого объекта «***Линейный градиент»***

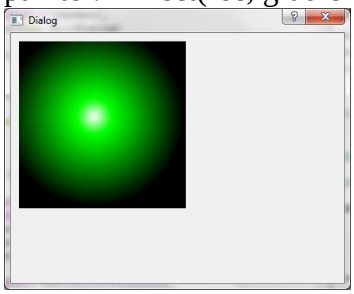
```
void Dialog::paintEvent(QPaintEvent *e)
{
//описание рисуемого объекта
}

// описание рисуемого объекта «Линейный градиент»
QPainter painter(this);
QLinearGradient gradient(50, 100, 300, 350); // создание градиента
gradient.setColorAt(0.0, Qt::white); // цвет градиента 1
gradient.setColorAt(0.2, Qt::green); // цвет градиента 2
gradient.setColorAt(1.0, Qt::black); // цвет градиента 3

QRect rec(10,10,200,200); // определение области заливки градиентом
painter.fillRect(rec, gradient); // заливка градиентом
```



В файле исходного кода dialog.cpp в описании paintEvent вставить код описания рисуемого объекта «***Радиальный градиент»***

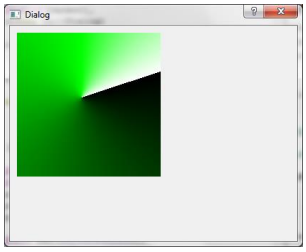
```
//описание рисуемого объекта «Радиальный градиент»
QRadialGradient gradient(100, 200, 130); // (центр, радиус, фокус)

gradient.setColorAt(0.0, Qt::white); // цвет градиента 1
gradient.setColorAt(0.2, Qt::green); // цвет градиента 2
```

```
gradient.setColorAt(1.0, Qt::black); // цвет градиента 3

QRect rec(10,10,200,200); // определение области заливки градиентом
painter.fillRect(rec, gradient); // заливка градиентом
```



В файле исходного кода dialog.cpp в описании paintEvent вставить код описания рисуемого объекта «***Конический градиент»***

```
//  описание рисуемого объекта «Конический градиент»

QPainter painter(this);
QConicalGradient gradient (100, 100, 18); //(центр1, центр2, угол)
gradient.setColorAt(0.0, Qt::white); // цвет градиента 1
gradient.setColorAt(0.2, Qt::green); // цвет градиента 2
gradient.setColorAt(1.0, Qt::black); // цвет градиента 3

QRect rec(10,10,200,200); // определение области заливки градиентом
painter.fillRect(rec, gradient); // заливка градиентом
```



Лабораторная работа № 19

Тема: Разработка приложения с использованием элементов управления.

Цель: Разработать программный продукт на Qt, позволяющий решить квадратное уравнение.

Время выполнения: 80 минут

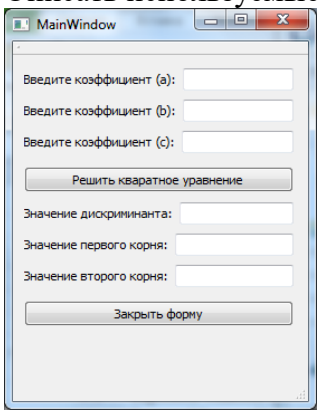
Проверяемые результаты обучения

Наименование объектов контроля и оценки	**Основные показатели оценки результата**	**Оценка (кол-во баллов)**
У 1. Осуществлять разработку кода программного модуля на современных языках программирования.	Работающий код программного модуля	3 балла
У 2. Создавать программу по разработанному алгоритму как отдельный модуль.	Осуществлена обработка событий	
У 3. Выполнять отладку и тестирование программы на уровне модуля.	Умение пользоваться отладчиком	

З 1. Основные этапы разработки программного обеспечения	Код программы разработан согласно этапам разработки программного обеспечения	
З 2. Основные принципы технологии структурного и объектно-ориентированного программирования	В коде программы соблюдены принципы технологии структурного и объектно-ориентированного программирования	
З 3. Основные принципы отладки и тестирования программных продуктов.	Отладка кода программного модуля на граничных точках	

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание.
Для заданных коэффициентов а, в, с найти корни уравнения вида: ax^2+вх + с =0.
Создать проект в комплексной среде Qt, позволяющий решить квадратное уравнение. Графический интерфейс представлен в виде окна с элементами управления для ввода коэффициентов и вывода корней уравнения. Кнопки используются для начала решения уравнения и выхода их приложения.
Описать используемые классы и обрабатываемые события.



Main.cpp
```
#include <QtGui/QApplication>
#include "mainwindow.h"

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```
mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <math.h>
#include <sstream>
```

```
using std::string;
using std::stringstream;

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::changeEvent(QEvent *e)
{
    QMainWindow::changeEvent(e);
    switch (e->type()) {
    case QEvent::LanguageChange:
        ui->retranslateUi(this);
        break;
    default:
        break;
    }
}

void MainWindow::on_pushButton_clicked()
{
    stringstream ss;
    float x1;
    float x2;
    float deskr;

    QString sx1;
    QString sx2;
    QString sdeskr;

    int a = ui->lineEdit->text().toInt();
    int b = ui->lineEdit_2->text().toInt();
    int c = ui->lineEdit_3->text().toInt();

    if(a == 0 && b == 0 && c == 0 && a != 0 && b != 0 && c != 0)
    {
        ui->label_7->setText("yPABHEHuE He KBADPATHOE");
    }

    deskr = (b * b) - (4 * a * c);
    sdeskr = QString::number(deskr);
    ui->lineEdit_4->setText(sdeskr);
```

```
    if(deskr < 0)
    {
        ui->label_7->setText("DECKPuMuHAHT MEHbLLIE HyJI9I");
    }

    if(deskr > 0)
    {
        x1 = ((b * (-1)) + sqrt(deskr)) / (2 * a);
        x2 = ((b * (-1)) - sqrt(deskr)) / (2 * a);
        sx1 = QString::number(x1);
        sx2 = QString::number(x2);
        ui->lineEdit_5->setText(sx1);
        ui->lineEdit_6->setText(sx2);
    }

    if(deskr == 0)
    {
        x1 = (b * (-1)) / (2 * a);
        sx1 = QString::number(x1);
        ui->lineEdit_4->setText(sx1);
    }
}

void MainWindow::on_pushButton_2_clicked()
{
    close();
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

namespace Ui {
    class MainWindow;
}

class MainWindow : public QMainWindow {
    Q_OBJECT
public:
    MainWindow(QWidget *parent = 0);
    ~MainWindow();

protected:
    void changeEvent(QEvent *e);

private:
    Ui::MainWindow *ui;

private slots:
    void on_pushButton_2_clicked();
    void on_pushButton_clicked();
```

```
};

#endif // MAINWINDOW_H
```

Лабораторная работа № 20

Тема: Методы и средства разработки технической документации программного продукта.
Цель: Разработать техническую документацию для программного продукта Qt.
Время выполнения: 120 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата
У 4. Оформлять документацию на программные средства.	Текстовый документ, выполненный согласно общим положениям о стандартах документирования программных средств
У 5. Использовать инструментальные средства для автоматизации оформления документации.	Использование текстового редактора.
З 3. Основные принципы отладки и тестирования программных продуктов	Изложение требований к тестированию, определение граничных точек, контроль задания на граничных точках, тестирование программы как чёрного ящика, как белого ящика, пошаговое тестирование, восходящее тестирование, нисходящее тестирование
З 4. Методы и средства разработки технической документации	Наличие титульного листа, постановки задачи, спецификаций, перечня входных и выходных данных, блок-схемы, сценария отладки программы, инструкции пользователя.

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание.
Для готового программного модуля, разработанного в Qt, создать документ, содержащий техническую документацию.
Методические указания:
Документация должна содержать необходимые сведения по установке, обеспечению надёжной работы продукта, справочное пособие для пользователя, демонстрационные версии, примеры документов, создаваемых при помощи данного программного продукта, обучающие программы.
Программная документация, включает:

- техническое задание (назначение, область применения программы, требования, предъявляемые к программе);
- текст программы (запись программы с необходимыми комментариями);
- описание программы (сведения о логической структуре и функционировании программы);
- пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений);
- эксплуатационные документы.

Программный документ "Пояснительная записка" составляется на стадии эскизного или технического проектов программы. Как правило, на стадии рабочего проекта не используется.

К эксплуатационным документам относят:

- описание применения (сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств);
- руководство системного программиста (сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения);
- руководство программиста (сведения для эксплуатации программы);
- руководство оператора (сведения для обеспечения общения оператора с вычислительной системой в процессе выполнения программы);
- описание языка (описание синтаксиса и семантики языка);
- руководство по техническому обслуживанию (сведения для применения тестовых и диагностических программ при обслуживании технических средств)

Основная часть программной документации составляется на стадии рабочего проекта. Необходимость того или иного документа определяется на этапе составления технического задания. Допускается объединять отдельные виды документов.

Эксплуатационный документ "Описание языка" включается в программную документацию, если разработанный программный продукт реализует некий язык программирования, управления заданиями, организации вычислительного процесса и т. п.

Эксплуатационный документ "Руководство по техническому обслуживанию" включается в программную документацию, если разработанный программный продукт требует использования тестовых или диагностических программ.

В техническое задание включают:

- введение (наименование, краткая характеристика области применения программы);
- основания для разработки (документы, на основании которых ведётся разработка, организация, утвердившая документы, дата утверждения, наименование и обозначение темы разработки);
- назначение разработки (функциональное и эксплуатационное назначение программы);
- требования к программе и программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приёмки.

Наиболее существенной частью технического задания является раздел "требования..." В этом разделе приводятся:

- требования к функциональным характеристикам (состав выполняемых функций, организация входных и выходных данных, временные характеристики);
- требования к надёжности (обеспечение устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа);
- требования к информационной и программной совместимости (требования к информационным структурам на входе и выходе, методам решения, исходным кодам, языкам программирования и программным средствам; требования к защите информации);
- требования к составу и параметрам технических средств;
- требования к программной документации.

Данный раздел может содержать требования к маркировке, упаковке, транспортировке и хранению, а также условия эксплуатации.
Кроме явно описанных в техническом задании требований, следует придерживаться общепринятых правил разработки программ с учётом выбранной парадигмы программирования:

8. Программа не должна содержать избыточные элементы (все элементы программы адекватны поставленной задаче: нет циклов, массивов и т. п. элементов, без которых можно обойтись).
9. Алгоритм должен быть структурирован: для функционального стиля программирования - адекватное разбиение на функции (процедуры), для объектно-ориентированного - адекватная иерархия классов. Каждая функция (метод класса) должна реализовывать ровно одно действие.
10. У функций (методов классов) должны быть параметры. Следует избегать использования в функциях глобальных переменных.
11. Программа должна аккуратно использовать память: работать с динамическими массивами, в ней не должно быть неиспользуемых блоков памяти, лишних переменных.
12. Должны проверятся диапазоны вводимых пользователем значений и параметров, передаваемых между модулями программы.
13. При использовании в программе каких-либо готовых компонент (библиотечных функций, классов) если функция или метод класса может завершиться неудачей, необходимо обязательно проверять это, не полагаясь на незначительность вероятности такого события.
14. Программа должна быть конфигурируема (важные параметры программы следует выделить в единый блок).

Текст программы

Текст программы представляет собой символическую запись на исходном или промежуточном языке или символическое представление машинных кодов. Текст программы оформляется моноширинным шрифтом (Courier, Lucida Console и т. п.):

8. Количество операторов на строчке должно быть равно 1.
9. Все операторы, входящие в составной оператор, должны быть сдвинуты вправо на одинаковое количество позиций, при этом операторные скобки (т. е. то, что ограничивает составной оператор), относящиеся к одному блоку, должны располагаться следующим образом: открывающая скобка должна находиться на той же строчке, что и оператор, открывающий блок, а закрывающая должна находиться в той же колонке, с которой начинается оператор, открывающий блок. Допускается располагать открывающую скобку на строке, следующей за оператором, открывающим блок, в той же колонке, с которой начинается этот оператор.
10. Строка исходного текста программы должна целиком располагаться в одной типографской строке (до 80 символов в зависимости от шрифта). Несоблюдение этого правила говорит о слишком большой вложенности блоков, что означает неудачный алгоритм или структуру программы. В таком случае рекомендуется переосмыслить структуру программы, ввести дополнительные функции, заменив какие-то большие части кода их вызовами, переделать алгоритм и т.п.
11. Если синтаксис языка позволяет, желательно отделять знаки операций пробелами от операндов. Как и в обычном тексте, после запятых должен следовать пробел.
12. Определения функций или логические части программы следует отделять друг от друга пустыми строками.

13. Идентификаторы (названия переменных, типов, подпрограмм) должны быть значимыми настолько, чтобы читающий текст программы мог понимать их смысл без присутствия рядом автора. При необходимости объявление переменной или типа может сопровождаться комментарием.
14. Текст программы должен содержать комментарии, отражающие функциональное назначение того или иного блока программы, структуру программы.

Описание программы

Документ "Описание программы" содержит:

- общие сведения (обозначение наименование программы, программное обеспечение, необходимое для функционирования программы, языки программирования, на которых написана программа);
- функциональное назначение (классы решаемых задач, сведения о функциональных ограничениях на применение);
- описание логической структуры (алгоритм программы, используемые методы, структура программы с описанием составных частей и связи между ними);
- используемые технические средства (типы ЭВМ и устройств, которые используются при работе программы);
- вызов и загрузка (способ вызова программы с соответствующего носителя данных);
- входные данные (характер, организация и предварительная подготовка входных данных, а также их формат, описание и способ кодирования);
- выходные данные (характер и организация выходных данных, а также их формат, описание и способ кодирования).

Описание логической структуры программы следует сопровождать блок-схемой программы.

Документ "Описание программы" может содержать также схемы данных, схемы взаимодействия программ, схемы ресурсов системы и проч., оформленные в соответствии с ГОСТ 19.701-90.

Описание применения

Документ "Описание применения" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (возможности, основные характеристики, ограничения области применения);
- условия применения (требования к техническим и программным средствам, общие характеристики входной и выходной информации, а также требования и условия организационного, технического и технологического характера);
- описание задачи (указываются определения задачи и методы её решения);
- входные и выходные данные.

Руководство системного программиста

Документ "Руководство системного программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания системным программистом. Документ состоит из следующих разделов:

- общие сведения о программе (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);
- структура программы (сведения о структуре, взаимосвязи между модулями программы и с другими программами);

- настройка программы (настройка на состав технических средств, выбор функций и т. п.);
- проверка программы (способы и методики проверки, контрольные примеры, методы прогона, результаты);
- дополнительные возможности;
- сообщения системному программисту (тексты сообщений, выдаваемых в ходе выполнения настройки, проверки программы, в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Руководство программиста

Документ "Руководство программиста" относится к эксплуатационным документам и включается в программную документацию, если разработанный программный продукт требует обслуживания программистом. Документ состоит из следующих разделов:

- назначение и условия применения программы (назначение и функции программы, сведения о технических и программных средствах, обеспечивающих выполнение данной программы);
- характеристики программы (временные характеристики, режимы работы, средства контроля правильности выполнения и т. п.);
- обращение к программе (способы передачи управления и параметров данных);
- входные и выходные данные (формат и кодирование);
- сообщения (тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Руководство оператора

Документ "Руководство оператора" относится к эксплуатационным документам и состоит из следующих разделов:

- назначение программы (информация, достаточная для понимания функций программы и её эксплуатации);
- условия выполнения программы (минимальный и/или максимальный набор технических и программных средств и т. п.);
- выполнение программы (последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы; описываются функции, форматы и возможные варианты команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды);
- сообщения оператору (тексты сообщений, выдаваемых оператору в ходе выполнения программы и описание действий, которые необходимо предпринять по этим сообщениям).

Лабораторная работа № 21

Тема:

Разработка пояснительной записки программного продукта приложения.

Цель:

Разработать пояснительную записку

Время выполнения: 120 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата

У 4. Оформлять документацию на программные средства.	Текстовый документ, выполненный согласно общим положениям о стандартах документирования программных средств
У 5. Использовать инструментальные средства для автоматизации оформления документации.	Использование текстового редактора.
З 3. Основные принципы отладки и тестирования программных продуктов	Изложение требований к тестированию, определение граничных точек, контроль задания на граничных точках, тестирование программы как чёрного ящика, как белого ящика, пошаговое тестирование, восходящее тестирование, нисходящее тестирование
З 4. Методы и средства разработки технической документации	Наличие титульного листа, постановки задачи, спецификаций, перечня входных и выходных данных, блок-схемы, сценария отладки программы, инструкции пользователя.

За верное выполнение работы выставляется – 3 балла.
За не полностью выполненную работу выставляется – 1 балл.
За невыполненную работу выставляется – 0 баллов.

Задание.
Для готового программного модуля создать пояснительную записку. Согласовать терминологию предметной области. Согласовать компьютерную терминологию.

Методические указания:

Документация должна содержать необходимые сведения по установке, обеспечению надёжной работы продукта, справочное пособие для пользователя, демонстрационные версии, примеры документов, создаваемых при помощи данного программного продукта, обучающие программы.
Программная документация, включает:

1. техническое задание (назначение, область применения программы, требования, предъявляемые к программе);
2. текст программы (запись программы с необходимыми комментариями);
3. описание программы (сведения о логической структуре и функционировании программы);
4. пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений);
5. эксплуатационные документы.

4. Пояснительная записка

Программный документ "Пояснительная записка" составляется на стадии эскизного или технического проектов программы. Как правило, на стадии рабочего проекта не используется.
Включает разделы:
1.Введение.
2.Назначение и область применения.
3.Технические характеристики.
4.Ожидаемые технико-экономические показатели.

5.Источники, используемые при разработке.

Введение содержит наименование программы и обозначение темы разработки, документы, на основе которых ведётся разработка.
В назначении и области применения указывают назначение программы, краткую характеристику области применения программы.
Технические характеристики содержат:
Постановка задачи на разработку программы, описание применяемых математических методов и различных ограничений, связанных с выбранным математическим аппаратом.
Описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи, возможного взаимодействия программы с другими программами.
Описание и обоснование выбора метода организации входных и выходных данных.
Описание и обоснование выбора состава технических и программных средств на основе проведённых расчётов и анализов, распределение носителей данных, которые использует программа.

3.2.3 Внеаудиторные самостоятельные работы

Внеаудиторная самостоятельная работа № 1

Текст задания

Подготовка реферата по одной из тем:
1.Новые концепции программирования.
2.Объектно-ориентированное программирование.
3.Однофайловые программы.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 2

Текст задания

Подготовка реферата по одной из тем:

- Технология сигналов и слотов
- Характеристики объектно-ориентированных языков. Объекты. Классы
- Программы с консольной графикой

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 3

Текст задания

Подготовка реферата по одной из тем:
Встроенные классы виджетов и диалоговых окон
Характеристики объектно-ориентированных языков. Наследование. Полиморфизм и перегрузка.
Отладка программ.
Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки	Выполнение разработки приложения поэтапно	2 балла

программного обеспечения; программирования;		
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 4

Текст задания

Подготовка реферата по одной из тем:

- Директивы препроцессора. Заголовочные файлы.
- Экранные заставки

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 5

Текст задания

Подготовка реферата по одной из тем:

- Подкласс QTableWidget.
- Центральный виджет . Структуры. Перечисления.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 6

Текст задания

Подготовка реферата по одной из тем:

- Подкласс QTableWidget
- Приоритеты операций языка С++.
- Операторы языка С++.

Время выполнения: 80 минут

Проверяемые результаты обучения

аименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 7

Текст задания

Подготовка реферата по одной из тем:

- Основные принципы отладки и тестирования программных продуктов.
- Подкласс QWidget. Объекты и классы.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 8

Текст задания

Подготовка реферата по одной из тем:

- Двойная буферизация.
- Структуры и классы.
- Классы, объекты и память.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 9

Текст задания

Подготовка реферата по одной из тем:

•Стековая компоновка

•Наследование. Базовые и производные классы.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки прамного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 10

Текст задания

Подготовка реферата по одной из тем:

- Прикрепляемые виджеты и панели инструментов
- Указатели. Управление памятью.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 11

Текст задания

Подготовка реферата по одной из тем:

- Переопределение обработчиков событий
- Виртуальные функции. Статические функции.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 12

Текст задания

Подготовка реферата по одной из тем:

- Установка фильтров событий.

- Потоковые классы.**Время выполнения:** 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 13

Текст задания

Подготовка реферата по одной из тем:
- Рисование при помощи QPanter. Преобразование рисовальщика.
- Создание многофайловых программ.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.

Внеаудиторная самостоятельная работа № 14

Текст задания

Подготовка реферата по одной из тем:
- Вывод на печатающее устройство.
- Шаблоны и исключения.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 15

Текст задания

Подготовка реферата по одной из тем:

- Технология “drag-and-drop”.
- Стандартная библиотека шаблонов (STL).

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 16

Текст задания

Подготовка реферата по одной из тем:
- Буфер обмена.
- Разработка объектно-ориентированного ПО

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 17

Текст задания

Подготовка реферата по одной из тем:
- Классов отображения элементов.
- Алгоритмы в стандартной библиотеке шаблонов.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.
За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 18

Текст задания

Подготовка реферата по одной из тем:

- Пользовательские модели.
- Пользовательские контейнеры библиотеки шаблонов.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.

За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 19

Текст задания

Подготовка реферата по одной из тем:

- Обобщённые алгоритмы.
- Множества и мультимножества. Отображения и мультиотображения.

Время выполнения: 80 минут

Проверяемые результаты обучения

аименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.

За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 20

Текст задания

Подготовка реферата по одной из тем:

- Строки, массивы байтов и объекты произвольного типа.
- Основы массивов. Массивы объектов. Массивы строк.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.

За невыполнение задания – 0 баллов

Внеаудиторная самостоятельная работа № 21

Текст задания

Подготовка реферата по теме:

Связь между процессами

Потоки и файлы. Потоковый ввод/вывод.

Обработка ошибок ввода/вывода. Реагирование на ошибки. Анализ ошибок.

Время выполнения: 80 минут

Проверяемые результаты обучения

Наименование объектов контроля и оценки	Основные показатели оценки результата	Оценка (кол-во баллов)
У1 Основные этапы разработки программного обеспечения; программирования;	Выполнение разработки приложения поэтапно	2 балла
У2 Основные принципы технологии структурного и объектно-ориентированного	Разработка приложения согласно требованиям структурного программирования	

За выполнение задания выставляется – 2 балла.

За невыполнение задания – 0 баллов

3.2.4 Задания промежуточной аттестации к МДК 01.02

3.2.4.1 Вопросы теоретического обучения

1. Дать определения понятий: программа, программное обеспечение, задача, приложение.
2. Дать определения понятий: предметная область, постановка задачи, алгоритм решения задачи.
3. Охарактеризовать деятельность системного программиста, прикладного программиста.
4. Охарактеризовать деятельность программиста аналитика, постановщика задач, администратора баз данных.
5. Охарактеризовать взаимодействие специалистов, связанных с созданием и эксплуатацией программ.
6. Дать определения понятий: утилитарные программы, программные продукты. Программный продукт.
7. Этапы создания ПП, сопровождения ПП.
8. Дать определение основных характеристик программ.
9. Дать определение жизненного цикла программного продукта
10. Как можно классифицировать методы создания программных продуктов в Qt.
11. Структура приложения Qt.
12. Что входит в прикладное программное обеспечение.
13. Назовите виды инструментальных средств Qt для разработки программных продуктов и дайте им краткую характеристику.
14. Что входит в состав системы программирования Qt.
15. Приведите примеры приемов надежного программирования приложения Qt.
16. Какие методы повышение производительности труда программиста вы знаете.
17. Приведите примеры и краткую характеристику программных продуктов для создания приложений.
18. Выбор и обоснование языка программирования Qt на примерах готовых программных продуктах.
19. Приведите примеры использования библиотек программ, встроенных функций.
20. Охарактеризуйте методы структурирования программ.

21. Дайте определение и характеристику объектно-ориентированного программирования.

22. По каким признакам происходит оценка эффективности программ.

23. Перечислите основные этапы отладки и сопровождения программных продуктов. Приведите примеры.

24. Охарактеризуйте ошибки программного обеспечения основные причины, источники и классификацию.

25. Для чего необходимо определение класса?

26. _____ имеет такое же отношение к _____, как стандартный тип данных к переменной этого типа.

27. В определении класса члены класса с ключевым словом private доступны:

любой функции программы;

- в случае, если вам известен пароль;
- методам этого класса;
- только открытым членам класса

28. Классы в объектно-ориентированном программировании

29. Истинно ли утверждение: поля класса должны быть закрытыми?

30. Операция точки (операция доступа к члену класса) объединяет следующие два элемента (слева направо):

- член класса и объект класса;
- объект класса и класс;
- класс и член этого класса;
- объект класса и член этого класса

31. Методы класса, определённые внутри класса, по умолчанию _____

32. Конструктор вызывается автоматически в момент _____ объекта.

33. Имя конструктора совпадает с именем _____.

34. Верно или неверно следующее утверждение: класс может иметь более одного конструктора с одним и тем же именем?

35. Методу класса всегда доступны данные:

объекта, членом которого он является;

класса, членом которого он является;

любого объекта класса, членом которого он является;

класса, объявленного открытым

36. Единственным формальным различием между структурами и классами в C++ является то, что _____

37. Пусть определены три объекта класса. Сколько копий полей класса содержится в памяти? Сколько копий методов функций?

38. Посылка сообщения объекту эквивалентна _____

39. Классы полезны потому, что:

- не занимают памяти, если не используются;
- защищают свои данные от доступа со стороны других классов;
- собирают вместе все аспекты, касающиеся отдельной вещи;
- адекватно моделируют объекты реального мира

40. Истинно ли следующее утверждение: существует простой, но очень точный метод, позволяющий представлять решаемую задачу в виде совокупности объектов классов?

41. В чём заключается разработка кода программного продукта согласно разработанному алгоритму в комплексной среде Qt.

42. Какова структура программы, разработанной в комплексной среде Qt.

43. Для чего делается выделение объектов и определение отношений между объектами.

44. На каком этапе программирования осуществляется проектирование классов.

45. Как выполняется компоновка программных компонентов и создание виджетов

46. Как производится отладка кода программного продукта с использованием отладчика комплексной среды Qt.

47. Реализация диалога в графическом пользовательском интерфейсе Qt.

48. Оценочное тестирование программного продукта.

49. Составление программной документации. Разработка пояснительной записки.

50. Разработка и назначение руководства пользователя, руководства системного программиста.

51. Разработка программного модуля с последовательными контейнерами. Ассоциативные контейнеры.

52. Установка среды Qt на ПК. Разработка кода программного модуля.

53. Подклассы QDialog.Технология сигналов и слотов.

54. Проектирование диалоговых окон. Динамические диалоговые окна. Встроенные классы виджетов и диалоговых окон.

55. Создание подкласса QMainWindow. Создание меню и панелей инструментов. Создание и настройка строки состояния.

56. Создание подкласса QMainWindow. Реализация меню File. Применение диалоговых окон. Сохранение настроек приложения. Экранные заставки. Основные этапы разработки программного обеспечения.

57. Основные принципы технологии структурного и объектно-ориентированного программирования, отладка и тестирование программы на уровне модуля.

58. Создание подкласса QTableWidget. Загрузка и сохранение. Реализация меню Edit.

59. Основные принципы отладки и тестирования программных продуктов, использующих настройки виджетов Qt и создание подкласса QWidget.

60. Интеграция пользовательских виджетов в QtDisigner. Двойная буферизация.

61. Области с прокруткой. Прикрепляемые виджеты и панели инструментов. Многодокументальный интерфейс.

62. Переопределение обработчиков событий. Установка фильтров событий.

63. Обработка событий во время продолжительных процессов.

64. Реализация пользовательских моделей. Методы и средства разработки технической документации

3.2.4.2 Вопросы практического обучения

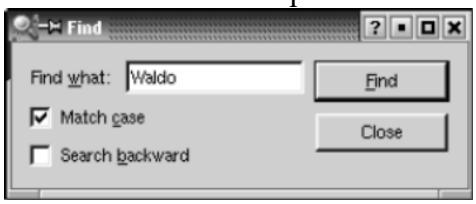
1. Напишите определение класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry().
2. Напишите оператор, создающий объект level1 класса, leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry().
3. Напишите оператор, который вызовет метод pry() объекта level1, имеющего тип класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry().
4. Напишите метод getcrow() для класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry(), который будет возвращать значение поля crowbar. Метод следует пределить внутри определения класса.
5. Напишите конструктор, который инициализирует нулевым значением поле crowbar класса leverage, включающего одно закрытое поле int с именем crowbar и одним

открытым методом с прототипом void pry(). Конструктор следует определить внутри определения класса.

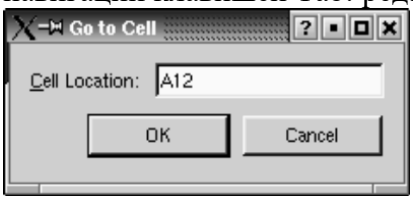
6. Предполагая, что метод getcrow() для класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry(), который возвращает значение поля crowbar, определён вне класса, объявите этот метод внутри класса.
7. Напишите метод getcrow() для класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry(), который будет возвращать значение поля crowbar. Метод следует определить вне класса.
8. Используя Qt Creator, создать проект. В списке Базовый класс выберите QMainWindow в качестве типа базового класса. Создать форму, содержащую пункты меню «Файл», «Сервис». Пункт «Файл» содержит подпункты «Создать», «Открыть», «Сохранить». На форме создать кнопку «Выход», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.
9. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. На форме создать кнопку «Выход», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.
10. Используя Qt Creator, создать проект. В списке Базовый класс выберите QDialog в качестве типа базового класса. На форме создать кнопку «Выход», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.
11. Используя Qt Creator, создать проект. В списке Базовый класс использовать QMainWindow — слой, в который можно добавлять любые дочерние виджеты. Разместить там поле для редактирования текста и кнопку для выхода, которая генерирует сигнал clicked() при нажатии пользователем кнопки.
12. Используя Qt Creator, создать проект. В списке Базовый класс выберите QDialog в качестве типа базового класса. На форме создать текстовый редактор и кнопку «Quit», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.

13. Используя Qt Creator, создать проект. В списке Базовый класс выберите QMainWindow в качестве типа базового класса. На форме создать 4 пункта меню и кнопку «Выход», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.
14. Используя Qt Creator, создать проект. В спискеБазовый класс выберите QWidget в качестве типа базового класса. На форме создать 3 кнопки. Разместить кнопки на форме горизонтально, используя **Qt layout**e.
15. Используя Qt Creator, создать проект. В спискеБазовый класс выберите QWidget в качестве типа базового класса. На форме создать 3 кнопки. Разместить кнопки на форме вертикально, используя **Qt layout**e.
16. Используя Qt Creator, создать проект. В спискеБазовый класс выберите QWidget в качестве типа базового класса. На форме создать 3 кнопки. Разместить кнопки на форме по сетке, используя **Qt layout**e.

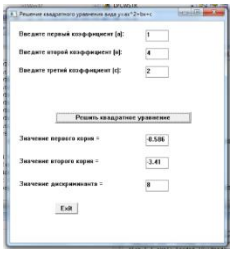
17. Используя Qt Creator, создать проект. В списке Базовый класс выберите QDialog в качестве типа базового класса. Создать и поместить в форму дочерние виджеты согласно прилагаемому рисунку. Создать компонент со своими сигналами и слотами. Скомпоновать по горизонтали и вертикали, подогнать размер.



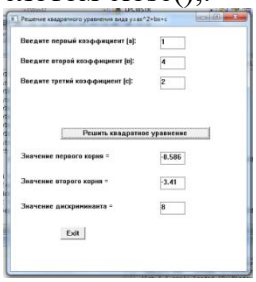
18. Используя Qt Creator, создать проект. В списке Базовый класс выберите QDialog в качестве типа базового класса. Создать и поместить в форму дочерние виджеты согласно прилагаемому рисунку. Создать компонент со своими сигналами и слотами. Скомпоновать по горизонтали и вертикали, подогнать размер. Установить порядок навигации клавишей Tab: редактор, кнопка «ОК», кнопка «Cancel»



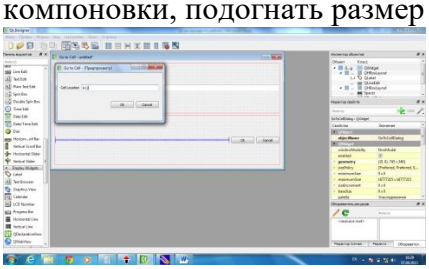
19. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. Создать и поместить в форму дочерние виджеты:Текстовая метка,Строка редактирования, 3 кнопки. Текстовую метку и строку редактирования скомпоновать по горизонтали. 3 кнопки скомпоновать по вертикали ниже предыдущих виджетов. Затем скомпоновать всё по сетке и подогнать размер. На форме создать кнопку «Выход», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.
20. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. Создать и поместить в форму дочерние виджеты согласно прилагаемому рисунку. На форме создать кнопку «Exit», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.



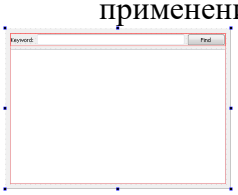
21. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. Создать и поместить в форму дочерние виджеты согласно прилагаемому рисунку. На форме создать кнопки «Решить квадратное уравнение» и “Exit”, используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.



22. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. Создать и поместить в форму дочерние виджеты: Текстовая метка,Строка редактирования, Горизонтальная распорка (space), 2 кнопки. Используя возможности редактора, создать горизонтальную и вертикальную компоновки, подогнать размер согласно рисунку:



23. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. Перетащите следующие виджеты на форму: Label (QLabel), Line Edit (QLineEdit), Push Button (QPushButton). На виджете **Label** введите текст **Keyword**. На виджете **Push Button** и введите текст **Find**. Измените значение свойства **objectName** на **findButton**. Скомпоновать виджеты по горизонтали. Перетащите виджет **Text Edit** (QTextEdit) на форму. Скомпоновать по вертикали для применения вертикальной компоновки



24. Укажите последовательность команд консоли для запуска приложения, созданного в Qt Designer или Qt Creator:
Запустить консоль: _____-> _____
Откроется окно:

В окне консоли ввести команды:
>_____
>_____
>_____
>_____
>_____
>_____

25. Сделать описание работы каждого оператора приложения hello:

```
#include <QApplication>
#include <QLabel>
int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   QLabel *label = new QLabel("Hello Qt!");
   label->show();
   return app.exec();
}
```

26. Сделать описание работы каждого оператора приложения Quit:

```
#include <QApplication>
#include <QPushButton>

int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   QPushButton *button = new QPushButton("Quit");
   QObject::connect(button, SIGNAL(clicked()),
                &app, SLOT(quit()));
   button->show();
   return app.exec();
}
```

27. Сделать описание работы каждого оператора приложения **:**

```
#include <QApplication>
#include <QTextEdit>
Int main(int argv, char  **args)
{
QApplication app(argv, args);
QTextEdit  textEdit;
textEdit.show();
return app.exec();
}
```

28. Используя Qt Creator, создать проект. В программе необходимо создать виджеты, второй и третий ключи сортировки которого не будут видны при выполнении программы, когда они не нужны. Сначала создаётся та часть, которая относится к первичному ключу, затем она дублируется дважды для получения вторичного и третичного ключа.

29. Создайте класс Int, имитирующий стандартный тип int. Единственное поле класса должно иметь тип int. Создайте методы, которые будут устанавливать значение поля, равным нулю, инициализировать его целым значением, выводить значение поля на экран и складывать два значения типа Int. Напишите программу, в которой будут созданы три объекта класса Int, два из которых будут инициализированы. Сложите два инициализированных объекта, присвойте результат третьему, а затем отобразите результат на экране.

30. Создайте класс с именем time, содержащий три поля типа int, предназначенные для хранения часов, минут и секунд. Один из конструкторов класса должен инициализировать поля нулевыми значениями, а другой конструктор – заданным набором значений. Создайте метод класса, который будет выводить значения полей на экран в формате 11:59:59, и метод, складывающий значения двух объектов типа time, передаваемых в качестве аргументов. В функции main()следует создать два

инициализированных значения, а результат присвойте третьему объекту и выведите его значение на экран. Где возможно, сделайте методы константными.

Критерии оценки программного продукта согласно показателям качества:

1. Показатели надежности программного продукта:
   − устойчивость функционирования;
   − работоспособность.
2. Показатели сопровождения:
   − структурность.
3. Простота конструкции.
4. Наглядность.
5. Повторяемость.
6. Показатели удобства применения.
7. Легкость освоения.

Критерии оценки:

«5»	«4»	«3»	«2»
Созданный программный продукт разработан в полном соответствии с показателями качества.	Созданный программный продукт имеет несоответствие по одному из показателей: простота конструкции.	Созданный программный продукт имеет несоответствие по двум показателям: простота конструкции, показатели удобства применения.	Созданный программный продукт не соответствует более 2 показателям.

4. Оценка по учебной практике

4.1. Общие положения

Целью оценки по учебной практике является оценка профессиональных и общих компетенций; практического опыта и умений.

Оценка по учебной практике выставляется на основании данных аттестационного листа (характеристики профессиональной деятельности студента на практике) с указанием видов работ, выполненных обучающимся во время практики, их объема, качества выполнения в соответствии с технологией и требованиями организации, в которой проходила практика.

4.2. Виды работ практики и проверяемые результаты обучения по профессиональному модулю

4.2.1. Учебная практика:

Таблица 3

Виды работ		Проверяемые результаты (ПК, ОК, ПО, У)
МДК.01.01		
Тема 1. Вводное занятие	***Виды работ*** Инструктаж о прохождении практики. Знакомство с программой практики и порядок её проведения, изучение правил внутреннего распорядка,	ОК 1 ПК 1.6 У1-У3 ПО 1-ПО 3

	знакомство с графиком работы студентов, ведения дневника практики, составление отчета. Инструктаж по технике безопасности, пожаробезопасности, производственной санитарии под роспись в журнале. Правила безопасности при работе с компьютером.	
Тема 2. Разработка алгоритма поставленной задачи и реализация его средствами автоматизированного проектирования.	***Виды работ*** Анализ поставленной задачи. Выбор методов и разработка основных алгоритмов решения задачи. Разработка технического задания. Примерные темы заданий для выполнения практической работы: • Обработка сообщений • Рисование геометрических фигур в окне • Вывод текста • Диалог с пользователем • Чтение и запись файлов в библиотеке Win32 API • Диалоговые окна • Растровая графика • Анимация • Библиотеки динамической компоновки DLL	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3
	Виды работ Разработка структуры и конкретных компонент разрабатываемого программного обеспечения, в том числе схемы алгоритмов, их общее описание, обоснование принятых технических решений. Математическая формализация. Построение информационной модели для решения поставленной задачи. Выделение объектов и процессов.	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3
	Виды работ Описание соотношений между характеристиками объектов моделирования. Системный анализ объектов моделирования поставленной задачи.	ОК 1-9 ПК 1.1 – 1.2
	Виды работ Реализация метода и основного алгоритма решения задачи методом последовательной детализации. Определение свойств входных и выходных данных поставленной задачи.	ОК 1-9 ПК 1.1 – 1.5 У1-У3 ПО 1-ПО 3
	Виды работ Анализ процесса обработки информации и выбор структур данных для её хранения	ОК 1-9 ПК 1.1 – 1.5 У1-У3 ПО 1-ПО 3
	Виды работ Построение алгоритма решения поставленной задачи средствами автоматизированного проектирования..	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3

Тема 3.Разработка кода программного продукта на основе готовой спецификации на уровне модуля.	***Виды работ*** Выбор технологии и среды программирования. Разработка структурной схемы программного продукта.	ОК 1-9 ПК 1.1 – 1.5 У1-У3 ПО 1-ПО 3
	Виды работ Анализ и уточнение требований к программному продукту. Применение технологии разработки многомодульных программ. Построение каркаса приложения.	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3
	Виды работ Проектирование интерфейса пользователя. Проектирование классов предметной области. Организация обработки сообщений.	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3
	Виды работ Построение графа диалога. Разработка форм ввода-вывода информации.	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3
	Виды работ Использование директив препроцессора для создания гибких и мобильных программ. Организация диалога с пользователем. Тестирование элементов управления.	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3
	Виды работ Работа с панелями инструментов**.** Чтение и запись файлов в библиотеке. Организация работы с файлами.	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3
	Виды работ Разработка кода программного продукта на языке С++ на уровне модуля.	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3
Тема 4. Использование инструментальных средств на этапе отладки программного продукта.	***Виды работ*** Выбор стратегии тестирования и разработка тестов. Отладка кода программного продукта, используя возможности отладчика.	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 3-ПО 4
	Виды работ Использование средств отладки, предоставляемых интерфейсом пользователя. Определение мест программы, в которых необходимо установить точки останова.	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
	Виды работ Использование команд меню Debug, Go для анализа значения переменных**.**	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
	Виды работ	ОК 1-9

	Настройка уровня предупреждений транслятора при компиляции программного кода.	ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
	Виды работ Использование программных средств отладки.	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
	Виды работ Использование директивы препроцессора #define для определения константы _DEBUG, используемой в директивах условной компиляции.	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
Тема 5. Проведение тестирования программного модуля по определенному сценарию.	***Виды работ*** Ручной контроль программного модуля. Проверка структуры программного модуля.	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
	Виды работ Использование отладочных функций для проверки допустимости значений объектов.	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
	Виды работ Структурное тестирование. Устранение утечки памяти. Исследование возможных причин утечки памяти.	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
	Виды работ Функциональное тестирование. Особенности отладки приложений, использующих шаблоны функций и классов.	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
	Виды работ Оценочное тестирование программного продукта. Анализ соответствия разработанного программного продукта постановке задачи.	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
Тема 6. Оформление документации на программные средства.	***Виды работ*** Составление программной документации. Определение сведений, необходимых для сопровождения и эксплуатации программного продукта. Разработка пояснительной записки, содержащей информацию о структуре и конкретных компонентах программного обеспечения, в том числе схемы алгоритмов, их общее описание, обоснование принятых технических решений.	ОК 1-9 ПК 1.6 У4-У5
	Виды работ Разработка спецификаций всех файлов программного продукта. Описание сведений о логической структуре и функционировании программы.	ОК 1-9 ПК 1.6 У4-У5
	Виды работ Разработка описания применения, содержащего	ОК 1-9 ПК 1.6

	сведения о назначении программного продукта, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств.	У4-У5
	Виды работ Разработка руководства системного программиста, содержащего сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения.	ОК 1-9 ПК 1.6 У4-У5
	Виды работ Разработка руководства по техническому обслуживанию, содержащего сведения для применения тестовых и диагностических программ при обслуживании технических средств.	ОК 1-9 ПК 1.6 У4-У5

МДК.01.02

Тема 1. Вводное занятие	***Виды работ*** Инструктаж о прохождении практики. Знакомство с программой практики и порядок её проведения, изучение правил внутреннего распорядка, знакомство с графиком работы студентов, ведения дневника практики, составление отчета. Инструктаж по технике безопасности, пожаробезопасности, производственной санитарии под роспись в журнале. Правила безопасности при работе с компьютером.	ОК 1 ПК 1.6 У1-У3 ПО 1-ПО 3
Тема 2. Разработка алгоритма поставленной задачи и реализация его средствами автоматизированного проектирования.	***Виды работ*** Выбор задания. Анализ постановки задачи. Анализ входных и выходных данных. Определение границ входных и выходных данных. Разработка алгоритма и блок-схемы поставленной задачи. Варианты заданий: Создание диалоговых окон. Создание главных окон. Графика 2D и 3D. Технология “drag-and-drop”. Управление компоновкой виджетов на форме. Обработка событий во время продолжительных процессов. Работа с каталогами.	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3
Тема 3. Разработка кода программного продукта на основе готовой спецификации на уровне модуля.	***Виды работ*** Разработка кода программного продукта согласно разработанному алгоритму в комплексной среде Qt. Разработка структуры программы. Выделение объектов и определение отношений между объектами. Проектирование классов. Компоновка программных компонентов. Создание виджетов.	ОК 1-9 ПК 1.1 – 1.2 У1-У3 ПО 1-ПО 3
Тема 4. Использование инструментальных средств на этапе отладки программного	***Виды работ*** Отладка кода программного продукта, используя возможности отладчика комплексной среды Qt. Реализация диалога в графическом пользовательском интерфейсе.	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4

продукта.		
Тема 5. Проведение тестирования программного модуля по определенному сценарию.	***Виды работ*** Ручной контроль программы. Структурное тестирование. Функциональное тестирование. Оценочное тестирование программного продукта.	ОК 1-9 ПК 1.3 – 1.5 У1-У3 ПО 3-ПО 4
Тема 6. Оформление документации на программные средства.	***Виды работ*** Составление программной документации. Разработка пояснительной записки, руководства пользователя, руководства системного программиста.	ОК 1-9 ПК 1.6 У4-У5

5. Контрольно-оценочные материалы для экзамена (квалификационного)

5.1. Паспорт
Назначение:
КОМ предназначен для контроля и оценки результатов освоения профессионального модуля **ПМ.01** РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНЫХ СИСТЕМ
по специальности СПО **230115** ПРОГРАММИРОВАНИЕ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

5.2. Задания для экзаменующегося:

Вариант №1
Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.
Инструкция:
1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:
– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;
- нормативной информацией и документами, используя Интернет-ресурсы.
3. Время выполнения задания – 3 часа.

Задание:
1. Дать определения понятий: программа, программное обеспечение, задача, приложение.

2. Охарактеризуйте методы структурирования программ.

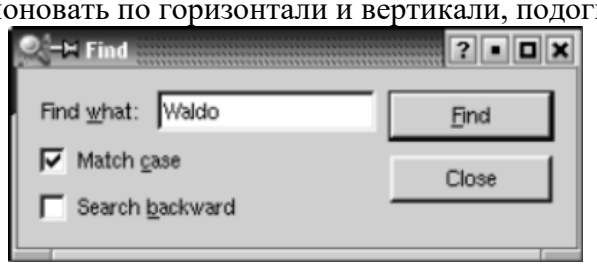
3. Напишите определение класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry().
class leverage
{
private: int crowbar;
public: void pry();
};
4. Создать простейшее приложение WinAPI, которое выводит второе окно при нажатии правой кнопки мыши в клиентской области окна функции.

Вариант № 2
Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.
Инструкция:
1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:
– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;
- нормативной информацией и документами, используя Интернет-ресурсы.
3. Время выполнения задания – 3 часа.

Задание:
1. Структура приложения Qt.
2. Операция точки (операция доступа к члену класса) объединяет следующие два элемента (слева направо):
   - член класса и объект класса;

- объект класса и класс;
- класс и член этого класса;
- объект класса и член этого класса

3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QDialog в качестве типа базового класса. Создать и поместить в форму дочерние виджеты согласно прилагаемому рисунку. Создать компонент со своими сигналами и слотами. Скомпоновать по горизонтали и вертикали, подогнать размер.



4. Создать простейшее приложение WinAPI, которое демонстрирует основные стили окон (окно верхнего уровня, всплывающее окно с родителем, всплывающее окно без родителя, дочернее окно).

Вариант № 3

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Дать определения понятий: предметная область, постановка задачи, алгоритм решения задачи.
2. Дайте определение и характеристику объектно-ориентированного программирования.
3. Напишите оператор, создающий объект level1 класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry().
4. Создать простейшее приложение WinAPI, использующее окна различных стилей (главное, всплывающее и дочернее). Зарегистрировать отдельные классы окон ("MainWindows", "PopupWindows" и "ChildWindows"), предусмотрев для каждого класса собственный цвет фона и собственную функцию окна.

Вариант № 4

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

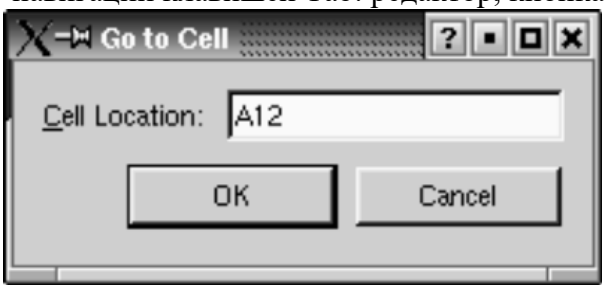
– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Что входит в прикладное программное обеспечение.
2. Перечислить функции, из которых состоит простейшее Windows - приложение.
3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QDialog в качестве типа базового класса. Создать и поместить в форму дочерние виджеты согласно прилагаемому рисунку. Создать компонент со своими сигналами и слотами. Скомпоновать по горизонтали и вертикали, подогнать размер. Установить порядок навигации клавишей Tab: редактор, кнопка «ОК», кнопка «Cancel»



4. Создать простейшее приложение WinAPI, использующее окна различных стилей (главное, всплывающее) зарегистрировать отдельные классы окон ("MainWindows", "PopupWindows "), предусмотрев для каждого класса собственный цвет фона и собственную функцию окна. Добавить в окно приложения вертикальную полосу прокрутки.

Вариант № 5

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Охарактеризовать деятельность системного программиста, прикладного программиста.
2. По каким признакам происходит оценка эффективности программ.
3. Напишите оператор, который вызовет метод pry() объекта level1, имеющего тип класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry().
4. Создать простейшее приложение WinAPI, использующее окон различных стилей (главное, всплывающее) зарегистрировать отдельные классы окон ("MainWindows", "PopupWindows "), предусмотрев для каждого класса собственный цвет фона и собственную функцию окна. Добавить в окно приложения горизонтальную полосу прокрутки. Включить стиль WS_HSCROLL в описание стиля создаваемого функцией CreateWindow окна

Вариант № 6

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

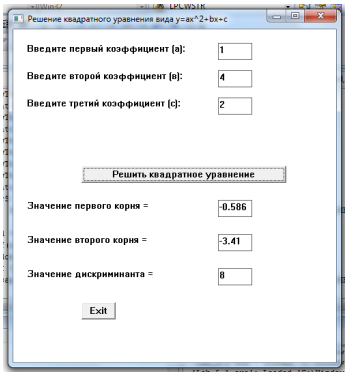
– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Что входит в состав системы программирования Qt.
2. Перечислить функции для создания окна в приложении Windows.
3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. Создать и поместить в форму дочерние виджеты согласно прилагаемому рисунку. На форме создать кнопку «Exit», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.



4. Создать простейшее приложение WinAPI, содержащее обработку сообщения WM_PAINT для вывода текста и геометрических фигур в окно.

Вариант № 7

Коды проверяемых профессиональных и общих компетенций:

ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Охарактеризовать деятельность программиста аналитика, постановщика задач, администратора баз данных.
2. Перечислите основные этапы отладки и сопровождения программных продуктов. Приведите примеры.
3. Напишите метод getcrow() для класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry(), который будет возвращать значение поля crowbar. Метод следует пределить внутри определения класса.
4. Создать простейшее приложение WinAPI, обрабатывающее сообщения от таймера.

Вариант № 8

Коды проверяемых профессиональных и общих компетенций:

ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

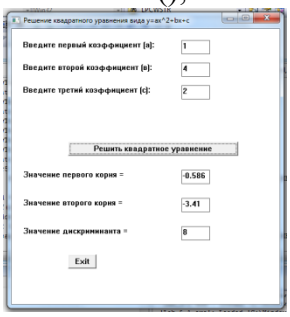
– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Приведите примеры приемов надежного программирования приложения Qt.
2. Перечислите типы окон в приложении Windows.
3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. Создать и поместить в форму дочерние виджеты согласно прилагаемому рисунку. На форме создать кнопки «Решить квадратное уравнение» и “Exit”, используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.



4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, text, label, button.

Вариант № 9

Коды проверяемых профессиональных и общих компетенций:

ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Охарактеризовать взаимодействие специалистов, связанных с созданием и эксплуатацией программ.
2. Охарактеризуйте ошибки программного обеспечения основные причины, источники и классификацию.
3. Напишите конструктор, который инициализирует нулевым значением поле crowbar класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry(). Конструктор следует определить внутри определения класса.
4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button для вычисления периметра и площадт прямоугольного треугольника по заданным длинам двух катетов a и b.

Вариант № 10

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Дать определения понятий: утилитарные программы, программные продукты.

   Программный продукт.

2. Дайте определение класса. Приведите пример класса и объектов типа этого класса.
3. Предполагая, что метод getcrow() для класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry(), который возвращает значение поля crowbar, определён вне класса, объявите этот метод внутри класса.
4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button для вычисления длины окружности и площади круга одного и того же заданного радиуса R.

Вариант № 11

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Этапы создания ПП, сопровождения ПП.
2. Способы взаимодействия дочерних окон с родительским окном в приложении WinAPI.
3. Напишите метод getcrow() для класса leverage, включающего одно закрытое поле int с именем crowbar и одним открытым методом с прототипом void pry(), который будет возвращать значение поля crowbar. Метод следует определить вне класса.
4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button для нахождения среднего арифметического кубов двух заданных чисел.

Вариант № 12

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Дать определение основных характеристик программ.
2. В определении класса члены класса с ключевым словом private доступны:
   любой функции программы;
   - в случае, если вам известен пароль;
   - методам этого класса;
   - только открытым членам класса

3. Используя Qt Creator, создать проект. В спискеБазовый класс выберите QMainWindow в качестве типа базового класса. Создать форму, содержащую пункты меню «Файл», «Сервис». Пункт «Файл» содержит подпункты «Создать», «Открыть», «Сохранить». На форме создать кнопку «Выход», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();
4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button для нахождения произведения цифр заданного четырехзначного числа.

Вариант № 13

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Дать определение жизненного цикла программного продукта
2. Классы в объектно-ориентированном программировании
3. Используя Qt Creator, создать проект. В спискеБазовый класс выберите QWidget в качестве типа базового класса. На форме создать кнопку «Выход», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.
4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button для вычисления суммы, разности, произведения и частного двух заданных чисел.

Вариант № 14

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Классификация методов создания программных продуктов в Qt.
2. Способы удаления окна с экрана в приложении WinAPI.

3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. На форме создать 3 кнопки. Разместить кнопки на форм, используя **Qt layout**е.
4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button и для заданной длины ребра куба найти площадь грани, площадь полной поверхности и объем этого куба.

Вариант № 15

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Назовите виды инструментальных средств Qt для разработки программных продуктов и дайте им краткую характеристику.
2. Указать, как вызывается конструктор в момент создания объекта.
3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. Создать и поместить в форму дочерние виджеты:Текстовая метка,Строка редактирования, 3 кнопки. Текстовую метку и строку редактирования скомпоновать по горизонтали. 3 кнопки скомпоновать по вертикали ниже предыдущих виджетов. Затем скомпоновать всё по сетке и подогнать размер. На форме создать кнопку «Выход», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.
4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button и для заданой стороны равностороннего треугольника найти площадь этого треугольника.

Вариант № 16

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

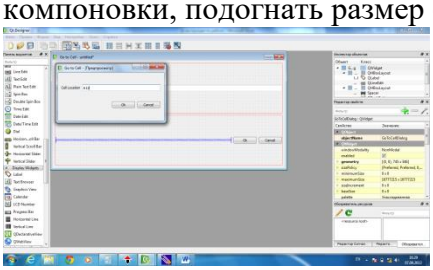
- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Какие методы повышение производительности труда программиста вы знаете.
2. Методу класса всегда доступны данные:
   объекта, членом которого он является;
   класса, членом которого он является;
   любого объекта класса, членом которого он является;
   класса, объявленного открытым

3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. Создать и поместить в форму дочерние виджеты: Текстовая метка, Строка редактирования, Горизонтальная распорка (space), 2 кнопки. Используя возможности редактора, создать горизонтальную и вертикальную компоновки, подогнать размер согласно рисунку:



4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button и для заданной стороны равностороннего треугольника найти его высоты.

Вариант № 17

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

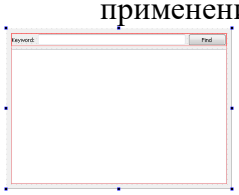
– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Выбор и обоснование языка программирования Qt на примерах готовых программных продуктах.
2. Понятие контекста устройства, как его получить в приложени WinAPI.
3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. Перетащите следующие виджеты на форму: Label (QLabel), Line Edit (QLineEdit), Push Button (QPushButton). На виджете **Label** введите текст **Keyword**. На виджете **Push Button** и введите текст **Find**. Измените значение свойства **objectName** на **findButton**. Скомпоновать виджеты по горизонтали. Перетащите виджет **Text Edit** (QTextEdit) на форму. Скомпоновать по вертикали для применения вертикальной компоновки



4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button и для заданной длины ребра куба найти объем этого куба.

Вариант № 18

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.

2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

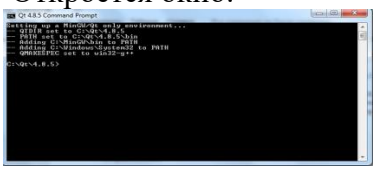
3. Время выполнения задания – 3 часа.

Задание:

1. В чём заключается разработка кода программного продукта согласно разработанному алгоритму в комплексной среде Qt.
2. Пусть определены три объекта класса. Сколько копий полей класса содержится в памяти? Сколько копий методов функций?
3. Укажите последовательность команд консоли для запуска приложения, созданного в Qt Designer или Qt Creator:

   Запустить консоль: _____

   Откроется окно:

   В окне консоли ввести команды:

   >_____

   >_____

   >_____

   >_____

   >_____

   >_____

4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button и для заданного основания a и b и угла α при большем основании a найти площадь равнобедренной трапеции.

Вариант № 19

Коды проверяемых профессиональных и общих компетенций:

ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Какова структура программы, разработанной в комплексной среде Qt.
2. Классы полезны потому, что:
   - не занимают памяти, если не используются;
   - защищают свои данные от доступа со стороны других классов;
   - собирают вместе все аспекты, касающиеся отдельной вещи;
   - адекватно моделируют объекты реального мира
3. Сделать описание работы каждого оператора приложения hello:

   #include <QApplication>

   #include <QLabel>

```
int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   QLabel *label = new QLabel("Hello Qt!");
   label->show();
   return app.exec();
}
```

4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button и для вычисления объёма цилиндра, имеющего высоту H и радиус основания R.

Вариант № 20

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Для чего делается выделение объектов и определение отношений между объектами.
2. На каком этапе программирования осуществляется проектирование классов.
3. Сделать описание работы каждого оператора приложения Quit:

```
#include <QApplication>
#include <QPushButton>

int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   QPushButton *button = new QPushButton("Quit");
   QObject::connect(button, SIGNAL(clicked()),
                    &app, SLOT(quit()));
   button->show();
   return app.exec();
}
```

4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button и для вычисления объёма конуса, который имеет высоту H и радиус основания R.

Вариант № 21

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Как выполняется компоновка программных компонентов и создание виджетов
2. Как производится отладка кода программного продукта с использованием отладчика комплексной среды Qt.
3. Сделать описание работы каждого оператора приложения **:**

```
#include <QApplication>
#include <QTextEdit>
Int main(int argv, char  **args)
{
QApplication app(argv, args);
QTextEdit  textEdit;
textEdit.show();
return app.exec();
}
```

4. Создать простейшее приложение WinAPI, использующее окна различных стилей (главное, всплывающее) зарегистрировать отдельные классы окон ("MainWindows", "PopupWindows "), предусмотрев для каждого класса собственный цвет фона и собственную функцию окна. Включить стиль WS_VSCROLL в описание стиля создаваемого функцией CreateWindow окна.

Вариант № 22

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

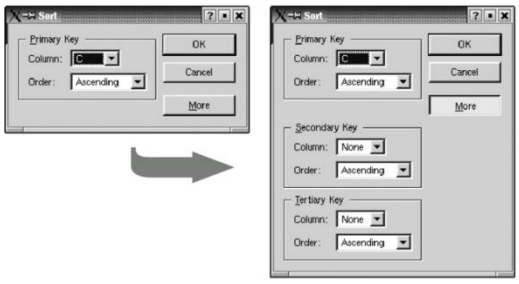
– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Реализация диалога в графическом пользовательском интерфейсе Qt.
2. Оценочное тестирование программного продукта.
3. Используя Qt Creator, создать проект. В программе необходимо создать виджеты, второй и третий ключи сортировки которого не будут видны при выполнении программы, когда они не нужны. Сначала создаётся та часть, которая относится к первичному ключу, затем она дублируется дважды для получения вторичного и третичного ключа.



4. Создать простейшее приложение WinAPI, использующее меню.

Вариант № 23

Коды проверяемых профессиональных и общих компетенций:

ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

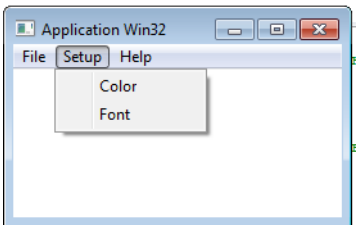
– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Установка среды Qt на ПК. Разработка кода программного модуля.
2. Составление программной документации.
3. Создайте класс Int, имитирующий стандартный тип int. Единственное поле класса должно иметь тип int. Создайте методы, которые будут устанавливать значение поля, равным нулю, инициализировать его целым значением, выводить значение поля на экран и складывать два значения типа Int. Напишите программу, в которой будут созданы три объекта класса Int, два из которых будут инициализированы. Сложите два инициализированных объекта, присвойте результат третьему, а затем отобразите результат на экране.
4. Создать простейшее приложение WinAPI, использующее окна меню и подменю:



Вариант № 24

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

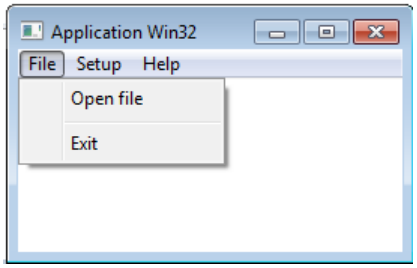
– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Подклассы QDialog.Технология сигналов и слотов.
2. Истинно ли следующее утверждение: существует простой, но очень точный метод, позволяющий представлять решаемую задачу в виде совокупности объектов классов?
3. Создайте класс с именем time, содержащий три поля типа int, предназначенные для хранения часов, минут и секунд. Один из конструкторов класса должен инициализировать поля нулевыми значениями, а другой конструктор – заданным набором значений. Создайте метод класса, который будет выводить значения полей на экран в формате 11:59:59, и метод, складывающий значения двух объектов типа time, передаваемых в качестве аргументов. В функции main()следует создать два инициализированных значения, а результат присвойте третьему объекту и выведите его значение на экран. Где возможно, сделайте методы константными.
4. Создать простейшее приложение WinAPI, использующее окна меню и подменю:



Вариант № 25

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Проектирование диалоговых окон. Динамические диалоговые окна. Встроенные классы виджетов и диалоговых окон.

2. Реализация пользовательских моделей. Методы и средства разработки технической документации

3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QWidget в качестве типа базового класса. На форме создать 3 кнопки. Разместить кнопки на форме вертикально, используя **Qt layout**e.

4. Создать простейшее приложение WinAPI, использующее окна меню и подменю:

Площадь	**Объём**
Прямоугольника	**Цилиндра**
Треугольника	**Конуса**
Трапеции	**Шара**
Круга	**Пирамиды**
	Параллелограмма

Вариант № 26

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Создание подкласса QMainWindow. Создание меню и панелей инструментов.

Создание и настройка строки состояния.

2. Обработка событий во время продолжительных процессов.

3. Используя Qt Creator, создать проект. В спискеБазовый класс выберите QWidget в качестве типа базового класса. На форме создать 3 кнопки. Разместить кнопки на форме по сетке, используя **Qt layout**e.
4. Создать простейшее приложение WinAPI, использующее окна меню и подменю:

Теоремы	Тригонометрия
Пифагора	Sin(x)
	Cos(x)
	Tg(x)
	Ctg(x)

Вариант № 27

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Создание подкласса QMainWindow. Реализация меню File. Применение диалоговых окон. Сохранение настроек приложения. Экранные заставки. Основные этапы разработки программного обеспечения.
2. Переопределение обработчиков событий. Установка фильтров событий.
3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QDialog в качестве типа базового класса. На форме создать кнопку «Выход», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close()

Вариант № 28

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Основные принципы технологии структурного и объектно-ориентированного программирования, отладка и тестирование программы на уровне модуля.
2. Создание подкласса QTableWidget. Загрузка и сохранение. Реализация меню Edit.

3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QMainWindow в качестве типа базового класса. На форме создать 4 пункта меню и кнопку «Выход», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close().
4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button и для вычисления объёма конуса, который имеет высоту H и радиус основания R.

Вариант № 29

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Основные принципы отладки и тестирования программных продуктов, использующих настройки виджетов Qt и создание подкласса QWidget.
2. Дайте определение и характеристику объектно-ориентированного программирования.
3. Используя Qt Creator, создать проект. В списке Базовый класс выберите QDialog в качестве типа базового класса. На форме создать текстовый редактор и кнопку «Quit», используя виджет QPushButton, который генерирует сигнал clicked() при нажатии пользователем кнопки. Сигнал кнопки clicked() связывается со слотом close();.

4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button и для заданного основания a и b и угла α при большем основании a найти площадь равнобедренной трапеции.

Вариант № 30

Коды проверяемых профессиональных и общих компетенций:
ПК 1.1 – ПК 1.6 , ОК 2 – ОК 9.

Инструкция:

1. Внимательно прочитайте задание.
2. Вы можете воспользоваться:

– справочной информацией, находящейся в разделах справочника и помощи в интегрированной среде разработки программ;

- нормативной информацией и документами, используя Интернет-ресурсы.

3. Время выполнения задания – 3 часа.

Задание:

1. Охарактеризуйте ошибки программного обеспечения, основные причины, источники и классификацию.

2. Реализация пользовательских моделей. Методы и средства разработки технической документации.
3. Используя Qt Creator, создать проект. В списке Базовый класс использовать QMainWindow — слой, в который можно добавлять любые дочерние виджеты. Разместить там поле для редактирования текста и кнопку для выхода, которая генерирует сигнал clicked() при нажатии пользователем кнопки.
4. Создать простейшее приложение WinAPI, с использованием элементов управления edit, button и для заданой стороны равностороннего треугольника найти площадь этого треугольника.

5.3. Пакет экзаменатора

5.3.1. Условия

Количество вариантов каждого задания/пакетов заданий для экзаменующегося: 30
Время выполнения каждого задания: 3 часа.
Оборудование: автоматизированное рабочее место обучающегося, точка доступа в Интернет, программное обеспечение общего и профессионального назначения.
Условия реализации программы ПРОФЕССИОНАЛЬНОГО МОДУЛЯ

Требования к минимальному материально-техническому обеспечению

Реализация программы **ПМ.01 Разработка программных модулей программного обеспечения компьютерных систем** предполагает наличие учебных кабинетов информатики и информационных технологий, лабораторий информатики и вычислительной техники.

Оборудование учебного кабинета и рабочих мест кабинета информатики и вычислительной техники: рабочие места по количеству обучающихся, компьютеры на рабочем месте учащихся с лицензионным программным обеспечением.

Технические средства обучения: персональные компьютеры с лицензионным программным обеспечением, мультимедийная система.

Программное обеспечение:
- Операционная система Windows XP,Vista,7;
- Среда Visual Studio 2010 Release Candidate или Visual Studio 2005/2008;
- Язык программирования Microsoft Visual C++
- Средство разработки Qt by Digiav4.8.5.(MinGW OpenSource);
- пакет прикладных программ Microsoft Office.

Оборудование лаборатории информационных технологий в профессиональной деятельности:

- автоматизированное рабочее место преподавателя;
- автоматизированные рабочие места обучающихся;
- компьютер с выходом в Интернет;
- интерактивная доска;
- компьютеры;
- программное обеспечение общего и профессионального назначения.

Основная литература:

4. Брайен, Керниган В. Язык программирования С : учебник / Брайен Керниган В., Деннис Ричи М. — Москва : Интуит НОУ, 2016. — 313 с. — URL: https://book.ru/book/918294

5. Александров, Э.Э. Программирование на языке C в Microsoft Visual Studio 2010 : курс лекций / Александров Э.Э., Афонин В.В. — Москва : Интуит НОУ, 2016. — 570 с. — URL: https://book.ru/book/918122

6. Алексеев, Е.Р. Программирование на языке C++ в среде Qt Creator : курс лекций / Алексеев Е.Р., Злобин Г.Г., Костюк Д.А., Чеснокова О.В., Чмыхало А.С. — Москва : Интуит НОУ, 2016. — 715 с. — URL: https://book.ru/book/918128

4.2.2 Электронные ресурсы:

4. ЭБС ИЗДАТЕЛЬСТВА "BOOK.RU" КОЛЛЕКЦИЯ СПО https://www.book.ru/

5. ЭБС ИЗДАТЕЛЬСТВА "ЮРАЙТ" https://urait.ru

6. ЭБС ИЗДАТЕЛЬСТВА "ЛАНЬ" https://e.lanbook.com

4.3 Дополнительная литература:

4. Макарова, Н.В. Основы программирования : учебник / Макарова Н.В., Нилова Ю.Н., Зеленина С.Б., Лебедева Е.В. — Москва : КноРус, 2021. — 451 с. — ISBN 978-5-406-03394-4. — URL: https://book.ru/book/936582

5. Фридман, А.Л. Язык программирования С : курс лекций / Фридман А.Л. — Москва : Интуит НОУ, 2016. — 218 с. — ISBN 978-5-9556-0017-8. — URL: https://book.ru/book/918295

6. Златопольский, Д.М. Программирование: типовые задачи, алгоритмы, методы : учебное пособие / Златопольский Д.М. 4-е изд. — Москва : Лаборатория знаний, 2020. — 224 с. — ISBN 978-5-00101-789-9. — URL: https://book.ru/book/936428

1.

5.3.2. Ход выполнения задания

Таблица

Коды проверяемых компетенций	Показатели оценки результата	Оценка (да/нет)
ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6, ОК 2, ОК 3, ОК 4, ОК 5, ОК 6, ОК 7, ОК 8, ОК 9.	1. Обращение в ходе задания к информационным источникам. 2. Рациональность распределения времени на выполнение задания: - ознакомление с заданием и планирование работы; - получение информации; - разработка и оформление программного продукта и технологической документации в соответствии со стандартами. 3. Соблюдение временных рамок выполнения задания. 4. Выполнение пользовательского интерфейса программного продукта на уровне модуля в соответствии с постановкой задачи. 5. Разработка компонент программных модулей с использованием современных инструментальных средств и технологий. 6. Оптимизация программного продукта и выявление избыточности кода программного модуля. 7. Точность, правильность и полнота выполнения профессиональных задач. 8. Способность к самоанализу и коррекции результатов собственной работы.	

2) Подготовленный продукт:

Таблица

Коды	Показатели оценки результата	Оценка

проверяемых компетенций		(да/нет)
ПК 1.1, ПК 1.2, ПК 1.3, ПК 1.4, ПК 1.5, ПК 1.6, ОК 2, ОК 3, ОК 4, ОК 5, ОК 8, ОК 9	− выполнение спецификаций компонентов; − выполнение создания кода программного продукта на уровне модуля в соответствии готовых спецификаций; − выбор модели проектирования пользовательского интерфейса; − разработка и отладка компонент с использованием современных инструментальных средства и технологий; − выполнение тестирования качества разработки программных модулей с помощью разработанных тестовых наборов и сценариев; - определение ошибок в программном коде с использованием тестовых наборов; − выполнение оптимизации программного кода модуля; − использование инструментальных средств и графических языков спецификаций для создания компонент проектной и технической документации в соответствии со стандартами; − мотивированное обоснование выбора и применения методов и способов решения профессиональных задач; − точность, правильность и полнота выполнения профессиональных задач; − демонстрация способности принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность; − аргументированность собственного мнения в выборе решения; − обоснованность выбора информационных источников для решения профессиональных задач; − оперативность поиска и использования необходимой информации для качественного выполнения профессиональных задач, профессионального и личностного развития; − широта использования различных источников информации, включая электронные; − ответственность за результат выполнения заданий; − способность к самоанализу и коррекции результатов собственной работы; − качество, своевременность и полнота выполнения заданий внеаудиторной самостоятельной работы; − обоснованность постановки цели и задач самообразования; − проявление интереса к инновациям в области профессиональной деятельности.	

3) Устное обоснование результатов работы

Таблица

Коды проверяемых компетенций	Показатели оценки результата	Оценка (да/нет)
ОК 4	− широта использования различных источников информации, включая электронные; − обоснованность выбора информационных источников для решения профессиональных задач; − оперативность поиска и использования необходимой информации для качественного выполнения профессиональных задач, профессионального и личностного развития.	

Исходный текст программы WinAPI **Приложение 1**

```
// --- Обязательный включаемый файл
#include <windows.h>

// --- Описание функции главного окна
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM
lParam);

// --- Глобальные переменные
HINSTANCE hInst;                    // Дескриптор экземпляра приложения
char ClassName[]="Window";          // Название класса окна
char AppTitle[]="Application Win32"; // Заголовок главного окна

// --- Функция WinMain
int WINAPI WinMain(
        HINSTANCE hInstance,        // Дескриптор экземпляра приложения
        HINSTANCE hPrevInstance,    // В Win32 всегда равен NULL
        LPSTR lpCmdLine,            // Указатель на командную строку. Он позволяет
                                    // приложению получать данные из командной
строки.
        int nCmdShow                // Определяет, как приложение первоначально
                                    // отображается на дисплее: пиктограммой
                                    // (nCmdShow = SW_SHOWMINNOACTIVE)
                                    // или в виде открытого окна
                                    //(nCmdShow = SW_SHOWNORMAL).
        )
{
        WNDCLASS wc;  // Структура для информации о класса окна
        HWND hWnd;    // Дескриптор главного окна приложения
        MSG msg;      // Структура для хранения сообщения

        // Сохраняем дескриптор экземпляра приложения в глобальной переменной,
        // чтобы при необходимости воспользоваться им в функции окна.
        hInst=hInstance;

        // --- Проверяем, было ли приложение запущено ранее.
        // Воспользуемся функцией FindWindow, которая позволяет найти окно верхнего
        // уровня по имени класса или по заголовку окна:
        //      HWND FindWindow(LPCTSTR lpClassName, LPCTSTR lpWindowName);
        // Через параметр lpClassName передается указатель на текстовую строку, в которую
        // необходимо записать имя класса искомого окна. На базе одного и того же класса
        // можно создать несколько окон. Если необходимо найти окно с заданным
заголовком,
        // то имя заголовка следует передать через параметр lpWindowName. Если же
подойдет
        // любое окно, то параметр lpWindowName может иметь значение NULL.
        if((hWnd=FindWindow(ClassName, NULL))!=NULL)
        {
                // Пользователь может не помнить, какие приложения уже запущены,
```

```
                // а какие нет. Когда он запускает приложение, то ожидает, что на экране
                // появится его главное окно. Поэтому, если приложение было запущено
                // ранее, целесообразно активизировать и выдвинуть на передний план
                // его главное окно. Это именно то, к чему приготовился пользователь.
                if(IsIconic(hWnd)) ShowWindow(hWnd, SW_RESTORE);
                SetForegroundWindow(hWnd);

                // Найдена работающая копия - работа новой копии прекращается.
                return FALSE;
        }

        // --- Работающая копия не найдена - функция WinMain приступает к
инициализации.
        // Заполнение структуры WNDCLASS для регистрации класса окна.
        memset(&wc, 0, sizeof(wc));
        wc.lpszClassName=ClassName;                             // Имя класса окон
        wc.lpfnWndProc=(WNDPROC)WndProc;                        // Адрес оконной
функции
        wc.style=CS_HREDRAW|CS_VREDRAW;                                 // Стиль
класса окон
        wc.hInstance=hInstance;                                 // Экземпляр
приложения
        wc.hIcon=LoadIcon(NULL,IDI_APPLICATION);                // Пиктограмма для
окон
        wc.hCursor=LoadCursor(NULL,IDC_ARROW);                          // Курсор
мыши для окон
        wc.hbrBackground=(HBRUSH)GetStockObject(WHITE_BRUSH);   // Кисть для
окон
        wc.lpszMenuName=NULL;                                   // Ресурс меню окон
        wc.cbClsExtra=0;                                // Дополнительная память
        wc.cbWndExtra=0;                                        // Дополнительная
память

        // Регистрация класса окна.
        RegisterClass(&wc);

        // Создаем главное окно приложения.
        hWnd=CreateWindow(
                ClassName,                      // Имя класса окон
                AppTitle,                       // Заголовок окна
                WS_OVERLAPPEDWINDOW,                    // Стиль окна
                CW_USEDEFAULT,                          // X-координаты
                CW_USEDEFAULT,                          // Y-координаты
                CW_USEDEFAULT,                          // Ширина окна
                CW_USEDEFAULT,                          // Высота окна
                NULL,                           // Дескриптор окна-родителя
                NULL,                           // Дескриптор меню окна
                hInst,                          // Дескриптор экземпляра приложения
                NULL);                          // Дополнительная информация
        if(!hWnd)
```

```
	{
		// Окно не создано, выдаем предупреждение.
		MessageBox(NULL,"Create: error",AppTitle,MB_OK|MB_ICONSTOP);
		return FALSE;
	}

	// Отображаем окно.
	ShowWindow(hWnd, nCmdShow);

	// Обновляем содержимое клиентской области окна.
	UpdateWindow(hWnd);

	// Запускаем цикл обработки очереди сообщений. Функция GetMessage получает
	// сообщение из очереди, выдает false при выборке из очереди сообщения
WM_QUIT
	while(GetMessage(&msg, NULL, 0, 0))
	{
		// Преобразование некоторых сообщений, полученных с помощью
клавиатуры
		TranslateMessage(&msg);

		// Отправляем сообщение оконной процедуре
		DispatchMessage(&msg);
	}

	return msg.wParam;
}

// --- Функция окна
LRESULT CALLBACK WndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM
lParam)
{
	char *str="First Windows aplication";

	switch(msg)
	{
		// Необходимо обновить содержимое клиентской области окна.
		case WM_PAINT:
		{
			HDC hDC;
			PAINTSTRUCT ps;

			hDC=BeginPaint(hWnd,&ps);			// Получить контекст
окна
			TextOut(hDC,20,20,str,strlen(str));	// Нарисовать текст
			EndPaint(hWnd,&ps);				// Освободить
контекст окна
		}; break;

		// Нажата левая клавиша мыши в клиентской области окна.
```

```
		case WM_LBUTTONDOWN:
		{
			MessageBox(hWnd, "32-bit aplication", "Window",
				MB_OK|MB_ICONINFORMATION);

		}; break;

		// Пользователь удалил окно.
		case WM_DESTROY:
		{
			// Если данная функция является оконной функцией главного
окна, то
			// следует в очередь сообщений приложения послать сообщение
WM_QUIT
			PostQuitMessage(0);
		}; break;

		// Необработанные сообщения передаем в стандартную
		// функцию обработки сообщений по умолчанию.
		default: return DefWindowProc(hWnd, msg, wParam, lParam);
	}
	return 0l;
}
```