

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования «Саратовский государственный технический  
университет имени Гагарина Ю.А.»

Энгельсский технологический институт (филиал)



УТВЕРЖДАЮ  
Зам. директора по СПДО

**Методические указания  
по выполнению практических работ  
ПМ.01 Разработка модулей программного обеспечения для  
компьютерных систем**

по специальности:

09.02.07 Информационные системы и программирование

Методические указания  
рассмотрены на заседании  
предметной (цикловой) методической комиссии  
специальности 09.02.07  
«25» июня 2024 года, протокол № 11

Председатель ПЦМК  А.А. Сдобнова

Энгельс 2024

**ОРГАНИЗАЦИЯ - РАЗРАБОТЧИК:**

Энгельсский технологический институт (филиал) федерального государственного бюджетного образовательного учреждения высшего образования «Саратовский государственный технический университет имени Гагарина Ю.А.»

**РАЗРАБОТЧИК:** Зотова А.А., преподаватель спецдисциплин ОСПДО

## СОДЕРЖАНИЕ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА .....	3
1 ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ ЗАНЯТИЙ .....	4
2. ОБЩИЕ РЕКОМЕНДАЦИИ ОБУЧАЮЩЕМУСЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ .....	16
3. КРИТЕРИИ ОЦЕНИВАНИЯ ВЫПОЛНЕННЫХ РАБОТ .....	178
4. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ .....	277

## ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

По учебному плану в соответствии с рабочей программой на изучение **ПМ.01 Разработка модулей программного обеспечения для компьютерных систем** обучающимися предусмотрено аудиторных занятий – 648 часов, из них практических занятий – 254 часов. В методические указания включены 104 практические работы по темам курса.

Целью практических занятий по **ПМ.01 Разработка модулей программного обеспечения для компьютерных систем** является закрепление студентами теоретического материала по специальности и приобретение практического навыка.

**После проведения практических занятий студент должен:**

**Иметь практический опыт**

в разработке кода программного продукта на основе готовой спецификации на уровне модуля;

использовании инструментальных средств на этапе отладки программного продукта;

проведении тестирования программного модуля по определенному сценарию; использовании инструментальных средств на этапе отладки программного продукта;

разработке мобильных приложений.

**уметь**

осуществлять разработку кода программного модуля на языках низкого и высокого уровней;

создавать программу по разработанному алгоритму как отдельный модуль;

выполнять отладку и тестирование программы на уровне модуля;

осуществлять разработку кода программного модуля на современных языках программирования;

уметь выполнять оптимизацию и рефакторинг программного кода;

- оформлять документацию на программные средства.

**знать**

основные этапы разработки программного обеспечения;

основные принципы технологии структурного и объектно-ориентированного программирования;

способы оптимизации и приемы рефакторинга;

основные принципы отладки и тестирования программных продуктов.

**1 ПЕРЕЧЕНЬ ПРАКТИЧЕСКИХ ЗАНЯТИЙ**  
**МДК. 01.01 Разработка программных модулей**

Номер и тема раздела	Номер практического занятия	Наименование темы занятия	Кол-во часов (ауд.)	Форма представления результата
1	2	3	4	5
<b>Тема 1.2 Структурное программирование</b>	1	Практическое занятие №1. Составление и оформление алгоритмов линейной структуры (следование)	4	Программный код алгоритмов линейной структуры (следование)
	2	Практическое занятие №2. Составление и оформление алгоритмов разветвляющейся структуры (ветвление)	2	Программный код алгоритмов разветвляющейся структуры (ветвление)
	3	Практическое занятие №3. Составление и оформление алгоритмов разветвляющейся структуры (ветвление)	2	
	4	Практическое занятие №4. Составление алгоритмов циклической структуры (повторение)	2	Программный код алгоритмов циклической структуры (повторение)
	5	Практическое занятие №5. Анализ сложности алгоритма суммирования членов последовательности, вычисления факториала	2	Отчет о проделанной работе
	6	Практическое занятие №6. Анализ сложности алгоритмов нахождения максимального и минимального числа	2	Отчет о проделанной работе

Номер и тема раздела	Номер практического занятия	Наименование темы занятия	Кол-во часов (ауд.)	Форма представления результата
	7	Практическое занятие №7. Алгоритмы вычисления бесконечных сумм	2	Программный код алгоритмов вычисления бесконечных сумм
	8	Практическое занятие №8. Алгоритмы поиска делителя натурально числа	2	Программный код алгоритмов поиска делителя натурально числа
	9	Практическое занятие №9. Алгоритм сортировки вставками	2	Программный код сортировки вставками
	10	Практическое занятие №10. Алгоритм сортировки методом «Пузырька»	2	Программный код сортировки методом «Пузырька»
	11	Практическое занятие №11. Алгоритм сортировки выбором	2	Программный код сортировки выбором
	12	Практическое занятие №12. Алгоритм последовательного поиска	2	Программный код последовательного поиска
	13	Практическое занятие №13. Алгоритм бинарного поиска	2	Программный код бинарного поиска
	14	Практическое занятие №14. Алгоритм блочного поиска	2	Программный код алгоритмов

Номер и тема раздела	Номер практического занятия	Наименование темы занятия	Кол-во часов (ауд.)	Форма представления результата
				блочного поиска
	15	Практическое занятие №15. Жадные алгоритмы	4	Программный код Жадных алгоритмов
	16	Практическое занятие №16 Составление и оформление рекурсивных алгоритмов	4	Программный код рекурсивных алгоритмов
	17	Практическое занятие №17 Составление и оформление рекурсивных алгоритмов	4	
	18	Практическое занятие №18 Составление эвристических алгоритмов.	6	Программный код эвристических алгоритмов
Тема 1.3 Объектно-ориентированное программирование	19	Практическое занятие №19 Работа с классами.	2	Программный код класса
	20	Практическое занятие №20 Перегрузка методов.	2	Программный код исходного и конечного вариантов метода
	21	Практическое занятие №21 Определение операций в классе.	2	Программный код операций в классе
	22	Практическое занятие №22 Создание наследованных классов	2	Программный код исходного и класса-наследника
	23	Практическое занятие №23 Работа с объектами через интерфейсы.	2	Форма для работы с объектами

Номер и тема раздела	Номер практического занятия	Наименование темы занятия	Кол-во часов (ауд.)	Форма представления результата
	24	Практическое занятие №24 Использование стандартных интерфейсов.	2	Форма для работы с объектами
	25	Практическое занятие №25 Работа с типом данных структура.	2	Программный код обработки структуры
	26	Практическое занятие №26 Коллекции. Параметризованные классы.	2	Программный код для работы с коллекцией
	27	Практическое занятие №27 Использование регулярных выражений	1	Программный код работы с регулярными выражениями
	28	Практическое занятие №28 Операции со списками.	1	Программный код обработки списка
Тема 1.4 Паттерны проектирования	29	Практическое занятие №29 Использование структурных шаблонов уровня класса.	2	Фрагмент кода приложения, реализующий шаблон
	30	Практическое занятие №30 Использование поведенческих шаблонов.	2	
	31	Практическое занятие №31 Использование порождающих шаблонов.	2	
	32	Практическое занятие №32 Использование структурных шаблонов уровня архитектуры.	2	
	33	Практическое занятие №33 Использование	2	



Номер и тема раздела	Номер практического занятия	Наименование темы занятия	Кол-во часов (ауд.)	Форма представления результата
Тема 1.5. Событийно-управляемое программирование		поведенческих шаблонов уровня архитектуры		
	34	Практическое занятие №34 Использование шаблонов интеграции	2	
	35	Практическое занятие №35 Разработка приложения с использованием текстовых компонентов	2	Фрагмент кода приложения реализующий использование текстовых компонентов
	36	Практическое занятие №36 Разработка приложения с несколькими формами.	2	Фрагмент кода приложения реализующий работу с несколькими формами
	37	Практическое занятие №37 Разработка приложения с не визуальными компонентами.	2	Фрагмент кода приложения реализующий работу с не визуальными компонентами
	38	Практическое занятие №38 Разработка игрового приложения.	2	Фрагмент кода приложения реализующий простейшую игру
	39	Практическое занятие №39	2	Фрагмент

Номер и тема раздела	Номер практического занятия	Наименование темы занятия	Кол-во часов (ауд.)	Форма представления результата
		Разработка приложения с анимацией.		кода приложения реализующий анимацию
Тема 1.6 Оптимизация и рефакторинг кода	40	Практическое занятие №40 Оптимизация кода.	2	Исходный и оптимизированный программный код
	41	Практическое занятие №41 Рефакторинг кода	2	Исходный программный код и программный код после рефакторинга
Тема 1.7 Разработка пользовательского интерфейса	42	Практическое занятие №42 Разработка интерфейса пользователя.	2	Программный код элемента интерфейса пользователя
Тема 1.8 Основы ADO.Net	43	Практическое занятие №43 Создание приложения с БД	1	Программный модуль для обработки базы данных
	44	Практическое занятие №44 Создание запросов к БД	1	
	45	Практическое занятие №45 Создание пользовательских форм	1	
	46	Практическое занятие №46 Создание хранимых процедур	1	
		<b>Всего</b>	<b>98</b>	

### МДК.01.02 Поддержка и тестирование программных модулей

Наименование раздела, темы	Номер практического занятия	Номер, название практической /лабораторной работы	Количество часов	Форма представления результата
1	2	3	4	5
Тема 1.9 Отладка и тестирование программного обеспечения	1	Практическое занятие №1 Тестирование «белым ящиком»	2	Отчет об ошибках, найденных в программном коде в процессе тестирования
	2	Практическое занятие №2 Альфа-тестирование отдельных модулей	2	
	3	Практическое занятие №3 Альфа-тестирование блоков связанных модулей	2	
	4	Практическое занятие №4 Проверка логики программы	2	
	5	Практическое занятие №5 Тестирование «черным ящиком»	2	
	6	Практическое занятие №6 Бета-тестирование	2	
	7	Практическое занятие №7 Установочное тестирование.	2	
	8	Практическое занятие №8 Модульное тестирование функции	2	
	9	Практическое занятие №9 Модульное тестирование класса	2	
	10	Практическое занятие №10 Интеграционное тестирование	2	
	11	Практическое занятие №11 Системное тестирование	2	
	12	Практическое занятие №12 Достижение и оценка надежности.	2	Оценка надежности
	13	Практическое занятие №13	2	Отчет от

Наименование раздела, темы	Номер практического занятия	Номер, название практической /лабораторной работы	Количество часов	Форма представления результата
		Регрессионное тестирование		ошибках
Тема 1.10 Документирование	14	Практическое занятие №14 Оформление документации на программные средства с использованием инструментальных средств	2	Документы, сформированные и оформленные согласно требованиям ГОСТ
	15	Практическое занятие №15 Оформление спецификации, формуляра	2	
	16	Практическое занятие №16 Формирование Технического задания	4	
	17	Практическое занятие №17 Формирование Текста программы	2	
	18	Практическое занятие №18 Формирование Руководства системного программиста	2	
	19	Практическое занятие №19 Формирование Руководства оператора (пользователя)	4	
	20	Практическое занятие №20 Внесение изменений в программные документы, выполненные печатным способом	2	
<b>Всего</b>			<b>44</b>	

Наименование раздела, темы	Номер практического занятия	Номер, название практической /лабораторной работы	Количество часов	Форма представления результата
<b>Тема 2.3</b>  <b>Классификация мобильных устройств. Архитектура мобильных устройств и их компонентов</b>	1	Практическое занятие № 1 Знакомство с платформой. Принципы работы.	4	Портфолио приложения
	2	Практическое занятие № 2 Создание эмуляторов и подключение устройств.	2	
	3	Практическое занятие № 3 Создание виртуального устройства первого приложения.	2	
	4	Практическое занятие № 4 Разработка мобильных приложений на примере XamarinStudio.	4	
<b>Тема 2.5</b>  <b>Проектирование и отладка мобильных приложений</b>	5	Практическое занятие № 5 Изучение и комментирование кода. Изменение элементов дизайна	4	Портфолио приложения
	6	Практическое занятие № 6 Обработка событий: подсказки. Обработка событий: цветовая индикация	4	
	7	Практическое занятие № 7 Подготовка стандартных модулей. Обработка событий: переключение между экранами	4	
	8	Практическое занятие № 8 Передача данных между модулями	4	
	9	Практическое занятие № 9 Разработка интерфейса для смартфонов. Принцип юзабилити	4	
	10	Практическое занятие № 10 Разметка активностей.	4	

Наименование раздела, темы	Номер практического занятия	Номер, название практической /лабораторной работы	Количество часов	Форма представления результата
		Компановка UI-элементов на экране приложения		
	11	Практическое занятие № 11 Тестирование и оптимизация мобильного приложения	4	
	12	Практическое занятие № 12 «Основы создания мобильных приложений»	2	
<b>Тема 2.6</b> <b>Основы работы в ОС Android</b>	13	Практическое занятие № 13 Архитектура платформы Android. Уровень ядра. Уровень библиотек	4	Портфолио приложения
	14	Практическое занятие № 14 Ресурсы в Android - приложениях.	2	
<b>Тема 2.9</b> <b>Использование баз данных и развертывание мобильных приложений</b>	15	Практическое занятие № 15 Расширения MS VisualStudio для разработки мобильных приложений	2	Портфолио приложения
	16	Практическое занятие № 16 Использование базы данных в мобильном приложении	2	
	17	Практическое занятие № 17 Дополнительных возможности при разработке мобильных приложений	2	
	18	Практическое занятие № 18 Отладка и развертывание мобильного приложения	2	
<b>Всего</b>			<b>56</b>	

**МДК 01.04 Системное программирование**

Наименование раздела, темы	Номер практического занятия	Номер, название практической /лабораторной работы	Количество часов	Форма представления результата
1	2	3	4	5
<b>Тема 2.11</b> <b>Файловые системы и функций символического ввода/вывода</b>		Практическое занятие № 1 Вывод на консоль сообщений и подсказок для пользователя	2	Портфолио приложения
	2	Практическое занятие № 2 Обработка ошибок	2	
	3	Практическое занятие № 3 Копирование нескольких файлов на стандартное устройство вывода	2	
	4	Практическое занятие № 4 Печать текущего каталога	2	
<b>Тема 2.12</b> <b>Реестр</b>	5	Практическое занятие № 5 Вывод списка разделов и содержимого реестра	2	Портфолио приложения
<b>Тема 2.13</b> <b>Обработка исключений</b>	6	Практическое занятие № 6 Обработка ошибок как исключений	2	Портфолио приложения
	7	Практическое занятие № 7 Использование обработчиков завершения для повышения качества программ	2	
	8	Практическое занятие № 8 Обработчик управляющих сигналов консоли	2	
<b>Тема 2.15</b> <b>Процессы</b>	9	Практическое занятие № 9 Управление процессами	2	Портфолио приложения
<b>Тема 2.16</b> <b>Библиотек и DLL</b>	10	Практическое занятие № 10 Создание библиотеки DLL	4	Портфолио приложения
<b>Тема 2.17</b> <b>Потоки и планирование выполнения</b>	11	Практическое занятие № 11 Использование потоков	2	Портфолио приложения
	12	Практическое занятие № 12 Синхронизация потоков	2	

Наименование раздела, темы	Номер практического занятия	Номер, название практической /лабораторной работы	Количество часов	Форма представления результата
1	2	3	4	5
я				
<b>Тема 2.18 Взаимодействие между процессами</b>	13	Практическое занятие № 13 Создание и открытие почтового ящика	4	Портфолио приложения
	14	Практическое занятие № 14 Создание, подключение и именованые каналы и почтовых ящиков	4	
<b>Тема 2.19 Сетевое программирование с помощью сокетов</b>	15	Практическое занятие № 15 Подготовка и подключение клиентских запросов соединения	4	Портфолио приложения
	16	Практическое занятие № 16 Подключение клиента к серверу	4	
	17	Практическое занятие № 17 Прием и отправка сообщений сокетом	4	
	18	Практическое занятие № 18 Клиент на основе сокета	4	
	19	Практическое занятие № 19 Сервер на основе сокетов	4	
	20	Практическое занятие № 20 Многопоточная DLL для обмена сообщениями через сокет	4	
		<b>Всего</b>	<b>56</b>	



## **2. ОБЩИЕ РЕКОМЕНДАЦИИ ОБУЧАЮЩЕМУСЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ**

1. Внимательно прочитайте задание, при необходимости повторите лекционный материал по конспектам и другим источникам, относящийся к теме практической/ лабораторной работы.

2. Ответьте на контрольные вопросы, если они предложены.

1. Подготовьте все необходимое для выполнения задания, рационально подготовьте рабочее место.

2. Продумайте ход выполнения работы.

3. Если ваша работа связана с использованием ИКТ, проверьте наличие и работоспособность программного обеспечения, необходимого для выполнения задания.

4. Если при выполнении практической работы применяется групповое или коллективное выполнение задания, старайтесь поддерживать в коллективе нормальный психологический климат, грамотно распределить роли и обязанности. Вместе проводите анализ организации и промежуточные результаты практической работы микрогруппы.

5. При выполнении практического задания соблюдайте правила техники безопасности и охраны труда.

6. В процессе выполнения практической работы обращайтесь за консультациями к преподавателю, чтобы вовремя скорректировать свою деятельность, проверить правильность выполнения задания.

7. По окончании выполнения практической работы составьте письменный или устный отчет в соответствии с теми методическими указаниями по оформлению отчета, которые вы получили от преподавателя или в методических указаниях.

8. Сдайте готовую работу преподавателю для проверки.

9. Участвуйте в обсуждении и оценке полученных результатов практической работы (общегрупповом или в микрогруппах).

### **Студент должен:**

- строго выполнять весь объем домашней подготовки, указанный в описаниях соответствующих практических работ;
- знать, что выполнению каждой работы предшествует проверка готовности студента, которая производится преподавателем;
- знать, что после выполнения работы бригада, которая назначается преподавателем на весь период работы, должна представить отчет о проделанной работе с обсуждением полученных результатов и выводов.

Процедура выставления окончательной оценки студенту по работе и порядок выполнения пропущенных работ по уважительным и неуважительным причинам следующая:

Объем работы исчисляется количеством полно и правильно выполненных заданий вне зависимости от порядка следования. Если задание не предполагает сохранения результатов в файле, то оценка выполнения производится по принципу вопрос-ответ или устное задание - выполненное действие. Процент выполнения рассчитывается для каждой работы исходя из общего числа заданий.

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

Для допуска студента к итоговой аттестации по МДК необходимо выполнение не менее 95% заданий.

## **МДК. 01.01 Разработка программных модулей**

## **МДК. 01.01 Разработка программных модулей**

### **Практическое занятие № 1**

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Составление и оформление алгоритмов линейной структуры (следование)

**Цель:** научиться составлять и оформлять алгоритмы линейной структуры

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

#### **Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

#### **Индивидуальные задания :**

1. Задано значение длины отрезка в метрах и миллиметрах. Найти ее величину в дюймах.
2. Задана длительность интервала времени в годах, месяцах и днях. Найти его величину в днях.
3. Задано количество команд, участвующих в чемпионате. Найти количество игр, если чемпионат проходит по круговой системе в два круга.
4. Задано значение длины отрезка в миллиметрах. Найти ее величину в футах и дюймах.
5. Задано значение угла в градусах, минутах и секундах. Найти его величину в радианах.
6. Задано значение длины отрезка в дюймах. Найти ее величину в метрах и миллиметрах.
7. Заданы коэффициенты квадратного уравнения. Найти его корни.

8. Заданы координаты центра окружности, ее радиус и значение некоторого угла. Найти координаты точки пересечения окружности и луча, исходящего из ее центра под заданным углом.
9. Задана высота прямоугольного равнобедренного треугольника, опущенная на гипотенузу. Найти длины сторон треугольника.
10. Заданы длины сторон прямоугольного равнобедренного треугольника. Найти высоту треугольника, опущенную на гипотенузу.
11. Задана сумма в рублях и копейках. Найти эквивалентную сумму в евро, долларах и центах.
12. Задана сумма денежных средств в рублях. Найти количество купюр и монет каждого номинала и их общее количество.
13. Заданы длительности двух интервалов времени и соответствующие скорости движения тела. Найти среднюю скорость движения тела.
14. Задана сумма средств в банке в рублях и копейках и годовой процент по вкладу. Найти сумму накопления за 5 лет в тех же единицах.
15. Задан год. Найти количество дней в году (365 или 366).
16. Задана скорость течения реки, ее ширина и скорость пловца в стоячей воде. Определить, под каким углом к фарватеру должен плыть пловец, чтобы пересечь реку под прямым углом.
17. Заданы корни квадратного уравнения. Найти его коэффициенты.
18. Заданы длины сторон треугольника. Найти его площадь.
19. Заданы длины двух сторон треугольника и угол между ними. Найти длину третьей стороны.
20. Заданы скорости двух тел одинаковой массы, движущихся под заданным углом. Найти скорость и угол движения тела относительно линии движения одного из них при абсолютно неупругом ударе.
21. Задан объем и высота резервуара и начальная скорость вытекания жидкости. Найти время, за которое вытечет вся жидкость.
22. Заданы количество сторон в основании правильной пирамиды, размер стороны и высота пирамиды. Найти объем пирамиды.
23. Заданы координаты двух точек, лежащих на некоторой прямой. Найти коэффициенты уравнения прямой, пересекающей данную в первой точке под прямым углом.
24. Заданы сопротивление, емкость, индуктивность и частота тока. Найти активное, реактивное и полное сопротивление их последовательного соединения.
25. Заданы размеры бильярдного стола, положение шара от угла у одной из стенок и угол удара. Найти расстояние от исходной позиции шара до точки первого касания той же стенки.
26. Заданы количество сторон в основании правильной пирамиды, размер стороны и высота пирамиды. Найти площадь поверхности пирамиды.

27. Заданы координаты двух соседних углов квадрата, повернутого на плоскости относительно осей координат на неизвестный произвольный угол.  
Найти координаты остальных углов.
28. Заданы длины двух сторон треугольника и угол между ними. Найти радиус окружности, вписанной в треугольник.
29. Заданы длины двух сторон треугольника и угол между ними. Найти длину третьей стороны.
30. Заданы емкости трех конденсаторов. Найти емкости всех возможных комбинаций их подключения.

#### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 2**

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Составление и оформление алгоритмов разветвляющейся структуры (ветвление)

**Цель:** научиться составлять и оформлять алгоритмы разветвляющейся структуры

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

#### **Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи

3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

### Индивидуальные задания

1. Заданы три числа. Определить, могут ли они являться длинами сторон треугольника и, если да, является ли этот треугольник равнобедренным и равносторонним.
2. Задан символ. Определить его принадлежность к одной из следующих групп: маленькая буква, большая буква, цифра, непечатный символ, прочее.
3. Заданы координаты центров и радиусы двух окружностей. Определить, лежит ли одна из них целиком внутри другой, пересекаются ли они.
4. Заданно трехзначное число. Определить, не применяя операцию отыскания остатка, делится ли число на 3, используя следующий признак делимости: сумма цифр делится на 3.
5. Задан номер группы ЭТИ СГТУ. Определить ее принадлежность к факультету, год поступления и курс, учитывая, что эта группа обучается в настоящее время.
6. Задан год. Найти главные спортивные события года (летняя и зимняя олимпиады, чемпионаты Европы и мира по футболу, чемпионат мира по хоккею с шайбой и т.п.).
7. Задан символ. Определить, может ли он являться цифрой в системах счисления с основаниями 2, 8, 10, 16.
8. Заданы координаты вершин треугольника. Вывести их в порядке обхода треугольника по часовой стрелке.
9. Заданы длины сторон двух треугольников. Определить, являются ли они подобными и, если да, коэффициент пропорциональности.
10. Задан радиус окружности и диагонали ромба. Определить, можно ли целиком разместить окружность внутри ромба и наоборот.
11. Задано число от 0 до 100, рассматриваемое как возраст человека. Вывести фразу вида: «Мне 21 год», «Мне 23 года», «Мне 25 лет».
12. Заданы координаты вершин прямоугольника и некоторой точки. Определить, лежит ли эта точка внутри прямоугольника, на его стороне или вне него.
13. Заданы размер прямоугольника и радиусы двух окружностей, определить, могут ли окружности быть размещены в прямоугольнике.
14. Заданы координаты вершин треугольников. Определить, является ли этот треугольник равнобедренным, равносторонним, прямоугольным.
15. Заданы координаты вершин двух прямоугольников. Определить их взаимное расположение: пересекаются, не пересекаются, касаются,

принадлежит.

16. Заданы коэффициенты квадратного уравнения. Найти его действительные корни, если они существуют.
17. Заданы длины сторон треугольника. Определить наименьший радиус окружности, в которой можно целиком разместить данный треугольник.
18. Заданы координаты вершин четырехугольника. Вывести их в порядке обхода по часовой стрелке.
19. Заданы фокусное расстояние выпуклой линзы и расстояние от предмета, находящегося на главной оптической оси, до линзы. Найти вид изображения и расстояние от линзы до него.
20. Заданы размеры трех прямоугольников. Определить, могут ли два из них быть размещены внутри третьего.
21. Заданы координаты короля и ладьи на шахматной доске. Определить, бьют ли фигуры друг друга.
22. Заданы координаты концов двух отрезков. Определить, пересекаются ли эти отрезки и, если да, найти координаты точки пересечения.
23. Заданы координаты черного короля, белых ферзя и ладьи на шахматной доске. Определить, имеет ли место мат.
24. Задан радиус окружности и стороны треугольника. Определить, можно ли целиком разместить окружность внутри треугольника и наоборот.
25. Заданы координаты короля и коня на шахматной доске. Определить, бьют ли фигуры друг друга.
26. Заданы координаты вершин четырехугольника. Определить его тип: прямоугольник, параллелограмм, трапеция, квадрат, произвольный.
27. Заданы координаты короля, слона и ладьи на шахматной доске. Определить, бьет ли ладья короля, находится ли король под защитой слона.
28. Заданы координаты начала и конца вектора на плоскости. Для произвольной точки на плоскости определить ее положение относительно вектора: слева, справа, впереди, позади, принадлежит.
29. Заданы координаты короля и слона на шахматной доске. Определить, бьют ли фигуры друг друга.
30. Заданы координаты вершин треугольника и некоторой точки. Определить, лежит ли эта точка внутри треугольника, на его стороне или вне него.

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя



### Практическое занятие № 3

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Составление и оформление алгоритмов разветвляющейся структуры (итерационные алгоритмы)

**Цель:** научиться составлять и оформлять алгоритмы разветвляющейся структуры

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

#### Последовательность выполнения работы

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

#### Индивидуальные задания

1. Задан интервал и шаг изменения аргумента. Вывести минимальное значение функции  $y = 3 \cdot \sin(2/x)$  на заданном интервале и соответствующее ему значение аргумента.
2. Задано целое число. Найти все простые числа, меньшие заданного.
3. Задано целое число. Используя средства стандартного ввода- вывода, изобразить на экране прямоугольный равнобедренный треугольник, катеты которого параллельны осям координат и равны заданному числу.
4. Задана точность. Определить требуемое количество членов разложения  $e = 2 + 1/2! + 1/3! + 1/4! + \dots$
5. Задан интервал и шаг изменения аргумента. Вывести минимальное значение функции  $y = 2x^2 + 5x - 7$  на заданном интервале и соответствующее ему значение аргумента.
6. Задано количество членов ряда Фибоначчи. Найти их значения и значение их суммы при  $a_0 = 0$ ,  $a_1 = 1$ .
7. Задан интервал и шаг изменения аргумента. Вывести максимальное значение функции  $y = 5 \cdot \cos(3x)$  на заданном интервале и соответствующее ему значение аргумента.

8. Ввести с клавиатуры географическую долготу и широту места и определить, в каком полушарии оно находится (в восточном или западном).
9. Задано количество цифр в номере билета. Определить количество «счастливых» билетов.
10. Задан интервал и шаг изменения аргумента. Вычислить значение площади под кривой  $y = 7 \cdot \sin(x)$  на заданном интервале.
11. Задано число. Найти его шестнадцатеричное представление.
12. Задан интервал и шаг изменения аргумента. Вычислить значение площади под кривой  $y = 2x + 2 \cdot \sin(x/3)$  на заданном интервале.
13. Задано число. Найти количество нулей в его двоичном представлении.
14. Задан радиус сферы и количество итераций. Найти объем сферы методом Монте-Карло и определить точность.
15. Заданы числа A и B. Представьте результат произведения этих чисел в 10-ной, 2-ной, 8-ной СС.
16. Задано число. Считая, что оно введено в восьмеричной системе счисления, найти его десятичное представление.
17. Задано целое число. Используя средства стандартного ввода-вывода, изобразить на экране окружность, диаметр которой равен заданному числу.
18. Заданы количество цифр числа и их сумма. Найти все числа, удовлетворяющие этому условию.
19. Заданы коэффициенты полинома третьей степени и точность. Найти корень уравнения методом Ньютона с заданной точностью.
20. Задано число. Определить является ли четным количество единиц в его двоичном представлении.
21. Задано целое положительное число. Найти все делители этого числа.
22. Задано количество разрядов числа, имеющих значение 1. Найти все возможные значения чисел.
23. Заданы два целых положительных числа X и Y. Найти значение  $X^Y$ , не используя операцию умножения.
24. Дано натуральное число. Определить, является ли оно четным или оканчивается цифрой 7.
25. Расстояния до двух ярчайших звезд северного полушария равны Сириус (созвездие Большого Пса) –  $8.14 \times 10^{12}$  км и Арктур (созвездие Волопаса) – 103 парсека (1 пс = 3.259 световых года). Определить, какая звезда находится дальше.

26. Известны массы и радиусы двух планет Венера  $m_v=4.86 \cdot 10^{27}$  г,  $r_v=6175$  км; Сатурн  $m_s=5.68 \cdot 10^{29}$  г,  $r_s=57750$  км. Определить, какая планета имеет наибольшее ускорение силы тяжести (формула для определения ускорения силы тяжести  $g=Gm/r^2$ , где универсальная гравитационная постоянная  $G=6.7 \cdot 10^{-8}$  г<sup>-1</sup>см<sup>2</sup>сек<sup>-2</sup>).
27. Ввести с клавиатуры текущее время и определить время суток (pm – с 0 до 12, am – с 12 до 24).
28. Ввести с клавиатуры координаты точки  $A(x,y)$ . Определить, лежит ли данная точка в четвертой четверти. Ответ вывести в виде сообщения.
29. Ввести с клавиатуры три целых числа ( $a, b, c$ ). Определить, являются ли они тройкой Пифагора ( $c^2 = a^2 + b^2$  или  $a^2=b^2+c^2$  или  $b^2=a^2+c^2$ ). Ответ вывести в виде сообщения.
30. Известны площади круга и квадрата. Определить: а) уместится ли круг в квадрат; б) уместится ли квадрат в круге

#### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

#### Практическое занятие № 4

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Составление алгоритмов циклической структуры (повторение)

**Цель:** научиться составлять и оформлять алгоритмы циклической структуры

**Материально-техническое и комплексно-методическое обеспечение:** для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи

алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

### **Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

### **Индивидуальные задания:**

1. Напечатать таблицу перевода расстояний из дюймов в сантиметры для значений длин от 1 до 20 дюймов  $1 \text{ дюйм} = 2,54 \text{ см}$ .
2. Вывести все четные числа кратные пяти в интервале от 2 до 100 включительно.
3. Даны натуральные числа от -500 до 500. Найти все трехзначные числа, у которых четные сотни
4. Определить сумму модулей всех нечетных, отрицательных чисел от -99 до 99.
5. Даны натуральные числа от 0 до 700. Найти все трехзначные числа, у которых нечетные сотни
6. Получить в порядке убывания все делители числа, введенного с клавиатуры.
7. Составьте программу определения наибольшего общего делителя двух натуральных чисел.
8. Составьте программу определения наименьшего общего кратного двух натуральных чисел.
9. Составьте программу, подсчитывающую количество цифр вводимого вами целого неотрицательного числа. Можно использовать операцию целочисленного деления
10. Даны числа от 1 до 1000 и число  $m$ . Вывести результат умножение куба нечетных сотен на число  $m$ .
11. Дано число  $n$  от 1 до 1000 и число  $m$ . Вывести результат квадрат разности числа  $n$  и число  $m$ .
12. Вычислить:  $1+2+4+8+\dots+2^{10}$  и  $(1+2) * (1+2+3) * \dots * (1+2+\dots+10)$ .
13. Даны числа от 1 до 1000 и число  $m$ . Вывести результат целочисленного деления нечетных сотен на число  $m$ .

14. Билет называют «счастливым», если в его номере сумма первых трех цифр равна сумме последних трех. Подсчитать число тех «счастливых» билетов, у которых сумма трех цифр равна 13. Номер билета может быть от 000000 до 999999.
15. Дано число  $n$  от 1 до 1000 и число  $m$ . Вывести результат квадрат целочисленного деления  $n$  на  $m$
16. Вводятся по очереди данные о росте  $N$  учащихся класса. Определить средний рост учащихся в классе
17. Составьте программу, суммирующую штрафное время команд при игре в хоккей. Выводить на экран суммарное штрафное время обеих команд после любого его изменения. После окончания игры выдать итоговое сообщение.
18. Дано натуральное число  $n$  ( $n < 9999$ ). Найти предпоследнюю цифру числа (в предположении, что  $n > 10$ ).
19. Даны числа от 1 до 1000 и число  $m$ . Вывести все остатки от деления четных сотен на число  $m$
20. Для заданного числа  $N$  составьте программу вычисления суммы  $S = 1 + 1/2 + 1/3 + 1/4 + \dots + 1/N$ , где  $N$  – натуральное число.
21. Каждая бактерия делится на две в течение одной минуты. В начальный момент имеется одна бактерия. Составьте программу, которая рассчитывает количество бактерий на заданное вами целое значение момента времени (15 минут, 7 минут и т.п.)
22. Составьте программу вывода на экран всех простых чисел, не превосходящих заданного  $N$ . Простым называется натуральное число больше единицы, имеющее только два делителя: единицу и само это число
23. Для чисел от 1 до 1000, найти сотни, в которых есть внутренние повторение (например, 122, 133, 144, 677 и т.д.)
24. Для чисел от 1 до 1000. Найти количество трехзначных чисел, все цифры которых одинаковы
25. Для чисел от 1 до 1000. Найти все нечетные сотни в которой есть повторение чисел
26. Для чисел от 1 до 1000, возвести в куб каждый третий десяток каждой второй сотни
27. Дано натуральное число  $n$  ( $n > 999$ ). Определить число сотен в нём
28. Даны натуральные числа от 0 до  $n$  ( $n < 99$ ) и число  $m$ . И найти квадрат первого числа больше  $m$

29. Дано натуральное число  $n$ . Найти все числа меньшие  $M_p$  числа Мерсенна. Число Мерсенна – это простое число, представленное в виде  $M_p = 2^p - 1$ , где  $p$  – тоже простое число.
30. Назовём натуральное число палиндромом, если его запись читается одинаково как с начала, так и с конца (пример: 4884, 393, 1, 22). Найти все меньшие 100 натуральных числа, которые при возведении в квадрат дают палиндром

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 5**

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Анализ сложности алгоритма суммирования членов последовательности, вычисления факториала

**Цель:** научиться выполнять анализ сложности алгоритма суммирования членов последовательности, вычисления факториала

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

#### **Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения

2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Выполнить анализ сложности алгоритмов, сделать вывод
6. Оформить отчет

**Этапы выполнения работы:**

**Задание 1.** Выполнить анализ сложности алгоритма суммирования членов последовательности согласно варианту

№ вар.	Сумма S
1	$1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots$
2	$\cos(x) + \frac{\cos(2x)}{2} + \frac{\cos(3x)}{3} + \dots$
3	$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots$
4	$\sin(x) - \frac{\sin(2x)}{2} + \frac{\sin(3x)}{3} - \dots$
5	$\cos(x) + \frac{\cos(3x)}{9} + \frac{\cos(5x)}{25} + \dots$
6	$1 + \frac{\cos(\frac{\pi}{4})}{1!}x + \frac{\cos(2\frac{\pi}{4})}{2!}x^2 + \dots$



№ вар.	Сумма S
7	$x - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots$
8	$x \sin\left(\frac{\pi}{4}\right) + x^2 \sin^2\left(\frac{\pi}{4}\right) + \dots$
9	$x + \frac{x^5}{5!} + \frac{x^9}{9!} + \frac{x^{13}}{13!} \dots$
10	$1 + \frac{\cos(x)}{1!} + \frac{\cos(2x)}{2!} + \dots$
11	$1 + \frac{3x^2}{1!} + \frac{5x^4}{2!} + \dots$
12	$\frac{x \cos(\frac{\pi}{3})}{1} + \frac{x^2 \cos(2\frac{\pi}{3})}{2} + \dots$
13	$\frac{x-1}{x+1} + \frac{1}{3}\left(\frac{x-1}{x+1}\right)^3 + \frac{1}{5}\left(\frac{x-1}{x+1}\right)^5 + \dots$
14	$-\cos(x) + \frac{\cos(2x)}{4} - \frac{\cos(3x)}{9} + \dots$
15	$\frac{x^3}{3} - \frac{x^5}{15} + \frac{x^7}{35} - \dots$
16	$\sin(x) + \frac{\sin(3x)}{3} + \frac{\sin(5x)}{5} + \dots$
17	$1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots$
18	$\frac{\cos(2x)}{3} + \frac{\cos(4x)}{15} + \frac{\cos(6x)}{35} + \dots$
19	$1 + \frac{2x}{1!} + \frac{(2x)^2}{2!} + \dots$
20	$1 + \frac{2}{1!}\left(\frac{x}{2}\right) + \frac{5}{2!}\left(\frac{x}{2}\right)^2 + \frac{10}{3!}\left(\frac{x}{2}\right)^3 \dots$
21	$x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots$
22	$1 - \frac{3}{2!}x^2 + \frac{5}{4!}x^4 - \frac{10}{6!}x^6 + \dots$
23	$-\frac{(2x)^2}{2!} + \frac{(2x)^4}{4!} - \frac{(2x)^6}{6!} + \dots$
24	$-(x+1)^2 + \frac{(x+1)^4}{2} - \frac{(x+1)^6}{3} + \dots$

№ вар.	Сумма S
25	$x - \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \dots$
26	$\frac{x \sin(\frac{\pi}{3})}{1} + \frac{x^2 \sin(2\frac{\pi}{3})}{2} + \dots$
27	$1 + \frac{\sin(\frac{\pi}{4})}{1!}x + \frac{\sin(2\frac{\pi}{4})}{2!}x^2 + \dots$
28	$x \cos(\frac{\pi}{4}) + x^2 \cos 2(\frac{\pi}{4}) + \dots$
29	$\frac{1}{2} \left( \frac{1}{x+1} \right)^2 + \frac{1}{4} \left( \frac{1}{x+1} \right)^4 + \dots$
30	$\sin(x) + \sin(3x^3) + \sin(5x^5) + \dots$

**Задание 2.** Написать программу вычисления факториала не используя стандартные функции. Выполнить анализ сложности алгоритма

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### Практическое занятие № 6

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Анализ сложности алгоритмов нахождения максимального и минимального числа

**Цель:** научиться выполнять анализ сложности алгоритмов нахождения максимального и минимального числа

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

### Последовательность выполнения работы

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Выполнить анализ сложности алгоритмов, сделать вывод
6. Оформить отчет

### Этапы выполнения работы:

**Задание 1.** Вывести значения функции согласно варианту. Найти максимальное и минимальное число на заданном интервале.

№ вар.	Функция	Интервал
1	$y = x^2 + \sin 5x$	[0.1;2.1]
2	$y = x^2 - \cos^2 \pi x$	[1;3]
3	$y = 1.8x^2 - \sin 10x$	[0.2;2.2]
4	$y = 2^x - 2x^2 - 1$	[2;4]
5	$y = x^2 - \cos^2(x+1)$	[0.1;2.2]
6	$y = x^3 - 4x^2 + 2$	[1;3]
7	$y = x - 3\cos^2(1.04x)$	[0.15;2.1]
8	$y =  e^x - 2  - x^2$	[1;3]
9	$y = x^3 + 3x^2 - 3$	[2;4]
10	$y = x^2 + 2\pi \cos \pi x$	[0.1;2.5]
11	$y = 5x^3 + 2x^2 - 15x - 6$	[1.3;3.4]
12	$y =  \lg x  - (x-2)^2$	[2;4.1]
13	$y = \sqrt{x} - 2\cos(0.5\pi x)$	[0.1;2.5]
14	$y = x^2 - x\pi \cos \pi x$	[0.1;2.1]
15	$y = x^3 - 7x - 7$	[1;3]
16	$y =  x^2 - 4  + 0.25x - 2$	[0.1;2.7]
17	$y = x^2 - \sin \pi x$	[0.1;2.9]
18	$y = \ln x^2 - x + 4$	[1;29]

№ вар.	Функция	Интервал
19	$y = x^3 - 6x^2 + 2$	[1.1;2.9]
20	$y = 3\sin\sqrt{x} + 0.25x - 3$	[1;3]
21	$y = 3\cos x -  x - 4  + 2$	[0;2]
22	$y = 0.25x^3 - 2.8x - 2$	[1.1;2.9]
23	$y = \ln x^2 - 1.8\sin x$	[1;3]
24	$y = x^2 + 4\sin \pi x$	[0.1;3]
25	$y = 0.5x^2 - 1 - \lg(x - 3)$	[1;2.9]
26	$y = \sqrt{1+x} - 3\cos x$	[0.1;3]
27	$y = \ln x^2 + x - 5$	[1;3]
28	$y = x^3 - 1.75x + 0.75$	[1;3]
29	$y = 0.5x - 1 - 2\cos(x + \frac{\pi}{4})$	[0;2]
30	$y = 3x - 2\ln x - 5$	[1.1;3.1]

**Задание 2.** Подсчитать сложность данного алгоритма

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### Практическое занятие № 7

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Алгоритмы вычисления бесконечных сумм

**Цель:** научиться составлять и оформлять алгоритмы вычисления бесконечных сумм

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

**Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

**Этапы выполнения работы:**

**Задание 1.** Вычислить бесконечную сумму ряда с заданной точностью  $\epsilon$  ( $\epsilon > 0$ ).

1.  $\sum_{i=1}^{\infty} \frac{1}{i^2}$

2.  $\sum_{i=1}^{\infty} \frac{1}{(i+1)^3}$

3.  $\sum_{i=2}^{\infty} \frac{(-1)^i}{i^2 - 1}$

4.  $\sum_{i=1}^{\infty} \frac{1}{i(i+1)}$

5.  $\sum_{i=2}^{\infty} \frac{5}{(i+1)(i-1)}$

6.  $\sum_{i=1}^{\infty} \frac{(-2)^{i+1}}{i(2i+1)}$

7.  $\sum_{i=1}^{\infty} \frac{2}{i!}$

8.  $\sum_{i=1}^{\infty} \frac{1}{(2i)!}$

9.  $\sum \frac{(-1)^i}{(2i-1)!}$

10.  $\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{2i!}$

11.  $\sum_{i=1}^{\infty} \frac{(-1)^{2i}}{i(i+1)(i+2)}$

12.  $\sum_{i=3}^{\infty} \frac{(-1)^{2i-1}}{i(i-1)(i-2)}$

13.  $\sum_{i=1}^{\infty} \frac{(-3)^{2i}}{3i!}$

14.  $\sum_{i=1}^{\infty} \frac{(-5)^{2i-1}}{5(2i-1)!}$

15.  $\sum_{i=1}^{\infty} \frac{1}{2^i}$

16.  $\sum_{i=1}^{\infty} \frac{1}{3^i + 4^i}$

17.  $\sum_{i=1}^{\infty} \frac{1}{5^i + 4^{i+1}}$

18.  $\sum_{i=1}^{\infty} \frac{(-1)^i}{2^{2i}}$

19.  $\sum_{i=1}^{\infty} \frac{(-1)^{i+1}}{3^{2i-1}}$

20.  $\sum_{i=1}^{\infty} \frac{1}{\sqrt{3^i}}$

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки

в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 8**

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Алгоритмы работы с одномерными массивами(повторение)

**Цель:** научиться составлять и оформлять алгоритмы работы с массивами

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

#### **Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

#### **Индивидуальные задания**

№ вар.	Задача
1	Ввести два целочисленных массива – по 10 элементов в каждом. Сформировать новый массив, на четных местах которого будут элементы с нечетными индексами из первого массива, а на нечетных – с четными индексами из второго.
2	Ввести массив, состоящий из 8 элементов (восемь двузначных чисел) целого типа. Получить новый массив, состоящий из цифр, находящихся в младших разрядах элементов исходного массива.
3	Ввести целочисленный массив, состоящий из 17-ти элементов (двузначные целые числа). Вычислить сумму цифр этого массива.
4	Ввести два массива действительных чисел, состоящих из 9 и 7 элементов. Сформировать третий массив из упорядоченных по возрастанию значений обоих массивов.
5	Ввести два массива $X$ и $Y$ , состоящих из 10-ти элементов целого типа. Сформировать массив $S$ , состоящий из одинаковых элементов исходных массивов.
6	Рассчитать значения 12-ти элементов массива $Y$ по формуле $y_i = i^2 - 2i + 19,3 \cos i$ . Вывести на экран этот массив и новый, разместив в нем первоначально элементы, значения которых меньше среднего арифметического, а потом остальные, не меняя их последовательности.
7	Дан массив вещественных чисел $Z(16)$ . Определить разность между суммой элементов с четными индексами и суммой элементов, индексы которых кратны трем.
8	В заданном целочисленном массиве $R(9)$ определить индекс наибольшего из нечетных по значению положительных элементов.
9	Ввести с клавиатуры массив $X$ , состоящий из 15 элементов целого типа. Рассчитать элементы массива $Y$ по формуле $y_i = \cos x_i^2 + 2,97 \lg^2 i^2$ . Сформировать третий массив из упорядоченных по убыванию значений обоих массивов.



№ вар.	Задача
10	<p>Ввести с клавиатуры массив <math>X</math>, состоящий из 17 элементов целого типа. Рассчитать элементы массива <math>Y</math> по формуле</p> $y_i = \begin{cases} (x_i)^3 - 7,5, & \text{если } \cos(x_i) > 0 \\ x_i^2 - 5e^{\sin(x_i)}, & \text{если } \cos(x_i) \leq 0 \end{cases}$ <p>Упорядочить массив <math>Y</math> по возрастанию, массив <math>X</math> по убыванию и сформировать новый массив <math>R</math>, элементами которого являются четные по индексу элементы массива <math>X</math> и <math>Y</math>.</p>
11	Ввести массив, состоящий из 9 элементов (девять двузначных чисел) целого типа. Получить новый массив, состоящий из сумм цифр элементов исходного массива.
12	Ввести массив, состоящий из 12 элементов действительного типа. Расположить элементы в порядке убывания. Определить количество происшедших при этом перестановок.
13	Ввести с клавиатуры целочисленный массив, состоящий из 11 элементов. Вычислить сумму нечетных по значению отрицательных элементов и заменить элементы кратные трем на эту сумму.
14	Ввести массив, состоящий из 14 элементов действительного типа. Поменять местами первую половину со второй. Определить количество произведенных при этом перестановок.
15	Дан массив вещественных чисел. Определить элемент массива (значение и индекс), который наиболее удален от заданного вещественного числа $S$ .
16	Ввести целочисленный массив, состоящий из 10 элементов. Определить сумму и количество элементов, расположенных до первого отрицательного числа.
17	Определить количество локальных минимумов в заданном числовом массиве. (Локальный минимум в числовом массиве – это последовательность трех рядом стоящих чисел, в которой среднее число меньше стоящих слева и справа от него).
18	Определить количество локальных максимумов в заданном числовом массиве. (Локальный максимум в числовом массиве – это последовательность трех рядом стоящих чисел, в которой среднее число больше стоящих слева и справа от него).
19	В заданном целочисленном массиве $Z(15)$ положительных, отрицательных и нулевых чисел определить сумму и вывести последовательность значений элементов, которые расположены между первым отрицательным и нулевым элементами.
20	В заданном числовом массиве определить и вывести индексы последовательностей чисел, которые монотонно убывают (каждое следующее число меньше предыдущего).



№ вар.	Задача
21	В заданном целочисленном массиве удалить элементы, которые встречаются более двух раз.
22	Ввести массив, состоящий из 10-ти элементов целого типа. Сформировать новый, расположив сначала все отрицательные элементы и нули, после чего - положительные, сохраняя порядок их следования.
23	Ввести массив $X$ , состоящий из 10-ти элементов целого типа. Вычислить элементы массива $Y$ по формуле $y_i = x_i^2 + 0,3$ $P = \frac{x_1 y_1 \cdot x_3 y_3 \cdot \dots \cdot x_9 y_9}{x_0 y_0 \cdot x_2 y_2 \cdot \dots \cdot x_8 y_8}$ и найти $P$ : Определить остаток от деления.
24	Ввести массив, состоящий из 10 элементов (десять двузначных чисел) целого типа. Получить новый массив, состоящий из разностей цифр элементов исходного массива.
25	Ввести массив, состоящий из 15 элементов целого типа. Упорядочить массив так, чтобы все отрицательные числа были расположены вначале по возрастанию, а все положительные – в конце по убыванию.
26	Даны два массива действительных чисел по 12 элементов в каждом. Заменить нулями те элементы первого массива, которые есть во втором.
27	Задан целочисленный массив. Определить количество участков массива, на котором элементы монотонно возрастают (каждое следующее число больше предыдущего).
28	Задан целочисленный массив. Определить остаток от деления суммы элементов с четными индексами на сумму элементов с нечетными индексами.
29	Задан целочисленный массив. Определить процентное содержание элементов, превышающих среднеарифметическое всех элементов массива.
30	Ввести два массива действительных чисел. Определить максимальные элементы в каждом массиве и поменять их местами.

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 9

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Алгоритмы работы с двумерными массивами (повторение).

**Цель:** Научиться работать с двумерными массивами

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

### Последовательность выполнения работы

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

### Индивидуальные задания:

№ вар.	Задание
1	В произвольной матрице - отсортировать по убыванию элементы последовательности, расположенные после второго отрицательного числа.
2	Необходимо заполнить двухмерный массив из 0 и 1. А после его вывода - массив должен иметь следующий вид: 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1 0

№ вар.	Задание
3	Необходимо заполнить двухмерный массив $A$ . После его вывода - массив должен иметь следующий вид: 01 02 03 04 12 13 14 05 11 16 15 06 10 09 08 07
4	Дан массив $A(n,m)$ . Удалить строки массива, не имеющие ни одного повторяющегося элемента.
5	Заполнить массив $3 \times 3$ числами по возрастанию, по спирали начиная с центра. 7 8 9 6 1 2 5 4 3
6	Элементы матрицы $A$ сделать с помощью генератора случайных чисел. Сделать новую матрицу $B$ , в которой удалить с матрицы $A$ ряд, в котором минимальный элемент среди элементов главной диагонали.
7	Составить программу, которая заполняет квадратную матрицу порядка $n$ натуральными числами $1, 2, 3, \dots, n^2$ , записывая их в нее "по спирали" против часовой стрелки.
8	Составить программу, которая заполняет квадратную матрицу порядка $n$ натуральными числами $1, 2, 3, \dots, n^2$ , записывая их в нее "по спирали" по часовой стрелке.
9	Дан двухмерный целочисленный массив $A(M,N)$ . Составить одномерный массив $B$ из номеров строк этого массива.
10	Написать программу, которая в матрице чисел $A(N,M)$ находит все элементы, превышающие по абсолютной величине заданное число $B$ . Подсчитать число таких элементов и записать их в массив $C$ .
11	Написать программу, которая в матрице чисел $A(N,M)$ находит все элементы, равные числу, введенному с клавиатуры. Подсчитать число таких элементов.
12	Задан двумерный массив $A[5,10]$ . Получить новую матрицу путем деления всех элементов исходной матрицы на ее наибольший по модулю элемент.
13.	Дан двумерный массив. Вставьте первую строку после строки, в которой находится первый встреченный минимальный элемент.
14.	Дан целочисленный массив $B[1..5, 1..5]$ . Вычислить произведение элементов этого массива, расположенных ниже левой диагонали.
15	Дан целочисленный массив $B[1..5, 1..5]$ . Вычислить сумму элементов этого массива, расположенных выше левой диагонали.
16	Дана целочисленная матрица размера $5 \times 5$ . Заменить в данной матрице все отрицательные элементы первой строки числом 0.

№ вар.	Задание
17	Дана целочисленная матрица размера $5 \times 5$ . Получить новую матрицу путем деления всех элементов данной матрицы на ее наибольший по модулю элемент.
18	Дана целочисленная прямоугольная матрица размера $M \times N$ . Отсортировать каждый столбец с четным номером по неубыванию, а каждый столбец с нечетным номером - по невозрастанию.
19	Дана целочисленная матрица размера $8 \times 5$ . Определить: а) сумму всех элементов второго столбца массива; б) сумму всех элементов 3-й строки массива.
20	Дана целочисленная прямоугольная матрица размера $M \times N$ . Сформировать одномерный массив, состоящий из элементов, лежащих в интервале $[1, 20]$ . Найти среднеарифметическое полученного одномерного массива.
21	Дана целочисленная прямоугольная матрица размера $M \times N$ . Сформировать одномерный массив, состоящий из элементов, лежащих в интервале $[1, 10]$ . Найти произведение элементов полученного одномерного массива.
22	Дана целочисленная квадратная матрица. Найти в каждой строке наибольший элемент и поменять его местами с элементом главной диагонали.
23	Дана целочисленная квадратная матрица. Указать столбец (назвать его номер), где минимальное количество элементов, кратных сумме индексов.
24	Дана целочисленная квадратная матрица. Найти сумму элементов матрицы, лежащих выше главной диагонали.
25	Определить, является ли данный квадратный массив симметричным относительно своей главной диагонали.
26	Определить, является ли данный квадратный массив не симметричным относительно своей главной диагонали.
27	Даны два числа $n$ и $m$ . Создайте двумерный массив <code>int A[n][m]</code> , заполните его таблицей умножения $A[i][j] = i * j$ и выведите на экран. При этом нельзя использовать вложенные циклы, все заполнение массива должно производиться одним циклом, например, <code>for(i=0; i&lt;n*m; ++i)</code> .
28	Дана матрица целых чисел размера $N \times M$ . Вывести номер строки, содержащей минимальное число одинаковых элементов.
29	Дана целочисленная квадратная матрица. Найти произведение элементов матрицы, лежащих ниже главной диагонали.
30	Дана матрица целых чисел размера $N \times M$ . Вывести номер строки, содержащей максимальное число одинаковых элементов.

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки



**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 10**

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Алгоритм сортировки методом «Пузырька», методом вставок

**Цель:** научиться составлять и оформлять алгоритм сортировки методом «Пузырька»

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

#### **Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

#### **Этапы выполнения работы:**

**Задание 1.** Отсортировать массив из практического задания №8 методом вставок

**Задание 2.** Отсортировать массив из практического задания №8 методом «Пузырька»

#### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на

половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 11**

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Алгоритм сортировки выбором

**Цель:** научиться составлять и оформлять алгоритм сортировки выбором

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

**Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

**Этапы выполнения работы:**

**Задание 1.** Отсортировать массив из практического занятия № 8 методом выбора

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## **Практическое занятие № 12**

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Алгоритм последовательного поиска

**Цель:** научиться составлять и оформлять алгоритм последовательного поиска

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

### **Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

### **Этапы выполнения работы:**

**Задание 1.** Отсортировать массив из практического занятия № 8 методом выбора. Найти максимальный, минимальный элементы массива.

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 13**

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Алгоритм бинарного поиска

**Цель:** научиться составлять и оформлять алгоритм бинарного поиска

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

#### **Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

#### **Этапы выполнения работы:**

**Задание 1.** В массиве из практического занятия № 8 с помощью алгоритма бинарного поиска найти минимальный и максимальный элементы

#### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя



## **Практическое занятие № 14**

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Алгоритм блочного поиска

**Цель:** научиться составлять и оформлять алгоритм блочного поиска

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

### **Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

### **Этапы выполнения работы:**

**Задание 1.** В массиве из практического занятия № 8 с помощью алгоритма блочного поиска найти минимальный и максимальный элементы массива

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 15

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Жадные алгоритмы

**Цель:** научиться составлять и оформлять жадные алгоритмы

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 360 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

### Последовательность выполнения работы

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

### Индивидуальные задания

1. Предположим, что трое ваших друзей, вдохновившись неоднократными просмотрами культового фильма “Ведьма из Блэр”, решили отправиться по Аппалачской тропе. Они хотят проходить как можно большее расстояние в дневное время, но по очевидным причинам не после наступления темноты. На карте обозначено множество хороших точек для разбивки лагеря. Чтобы выбрать место для остановки, они используют следующее правило: подходя к очередной возможной точке, они определяют, удастся ли им добраться до следующей точки до наступления темноты. Если время позволяет, то они продолжают идти, а если нет — останавливаются.
2. Есть купюры и монеты номиналами: 1, 5, 10, 50, 100, 1000, 5000. В банкомате неограниченное количество купюр каждого номинала. Мы хотим снять со счета  $n$  рублей. Нужно определить минимальное суммарное количество купюр и монет, которое может выдать банкомат, чтобы сумма получилась ровно  $n$ .
3. Ваши друзья открывают компанию, занимающуюся компьютерной безопасностью, и им необходимо получить лицензии на  $n$  разных криптографических продуктов. Согласно действующим требованиям, они могут получать не более одной лицензии в месяц.

Каждая лицензия в настоящее время стоит \$100, но со временем цена растет по экспоненциальному закону: в частности, лицензия  $j$  ежемесячно умножается на коэффициент  $r_j > 1$  (где  $r_j$  — заданный параметр). Таким образом, если лицензия  $j$  приобретается через  $t$  месяцев, она будет стоить  $100 \cdot r_j^t$ . Будем считать, что все скорости роста цены различны, то есть  $r_i \neq r_j$  для  $i \neq j$  (хотя все они начинаются с одной исходной цены \$100).

Вопрос: если компания может покупать не более одной лицензии в месяц, в каком порядке следует совершать покупки, чтобы общая потраченная сумма была минимальной?

Приведите алгоритм, который получает  $n$  скоростей роста цены  $r_1, r_2, \dots, r_n$  и вычисляет порядок покупки лицензий, чтобы общая сумма потраченных денег была минимальной. Время выполнения алгоритма должно быть полиномиальным по  $n$ .

4. Толик придумал новую технологию программирования. Он хочет уговорить друзей использовать ее. Однако все не так просто.  $i$ -й друг согласится использовать технологию Толика, если его авторитет будет не меньше  $a_i$  (авторитет выражается целым числом). Как только он начнет ее использовать, к авторитету Толика прибавится число  $b_i$  (попадаются люди, у которых  $b_i < 0$ ). Помогите Толику наставить на путь истинный как можно больше своих друзей. Входные данные

На первой строке входного файла INPUT.TXT содержатся два числа:  $n$  ( $1 \leq n \leq 1000$ ) — количество друзей у Толика, и первоначальный авторитет Толика. Следующие  $n$  строк содержат пары чисел  $a_i$  и  $b_i$ . Все числа целые, по модулю не больше 106.

Выходные данные

В выходной файл OUTPUT.TXT выведите число  $m$  — максимальное число друзей, которых может увлечь Толик, и затем  $m$  чисел — номера друзей в том порядке, в котором их нужно агитировать.

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки



**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 16-17**

**Тема раздела:** Структурное программирование

**Тема практического занятия:** Составление и оформление рекурсивных алгоритмов

**Цель:** научиться составлять и оформлять рекурсивные алгоритмы

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 720 минут

**Форма отчетности по занятию:** отчет по практическому занятию должен содержать: формулировку задания, блок-схему алгоритма, решение задачи алгоритмическом языке и на языке программирования C#, вывод о проделанной работе

#### **Последовательность выполнения работы**

1. Определить входные и выходные данные для решения задачи, сформулировать математическую модель решения
2. Составить блок-схему алгоритма решения задачи
3. Написать код на алгоритмическом языке
4. Написать код на языке программирования C#
5. Оформить отчет

#### **Индивидуальные задания:**

1. Составить рекурсивный алгоритм для вычисления цепной дроби: . Найти значение данной дроби при заданном натуральном  $n$  .
2. Разработать рекурсивный метод для перевода числа из десятичной системы счисления в двоичную
3. Даны первый член и разность арифметической прогрессии. Написать рекурсивный метод для нахождения  $n$ -го члена и суммы  $n$  первых членов прогрессии.
4. Даны первый член и знаменатель геометрической прогрессии. Написать рекурсивный метод для нахождения  $n$ -го члена и суммы  $n$  первых членов прогрессии.
5. Разработать рекурсивный метод, который по заданному натуральному числу  $N$

( $N \geq 1000$ ) выведет на экран все натуральные числа не больше  $N$  в порядке возрастания. Например, для  $N=8$ , на экран выводится 1 2 3 4 5 6 7 8.

6. Разработать рекурсивный метод, который по заданному натуральному числу  $N$  ( $N \geq 1000$ ) выведет на экран все натуральные числа не больше  $N$  в порядке убывания. Например, для  $N=8$ , на экран выводится 8 7 6 5 4 3 2 1.
7. Разработать рекурсивный метод для вывода на экран цифр натурального числа в прямом порядке. Применить эту процедуру ко всем числам из интервала от  $A$  до  $B$ .
8. Разработать рекурсивный метод для перевода числа из десятичной системы счисления в двоичную.
9. Разработать рекурсивный метод для перевода числа из двоичной системы счисления в десятичную.
10. Разработать рекурсивный метод для вывода на экран всех делителей заданного натурального числа  $n$ .
11. Задан треугольник размером  $m \times n$ , где  $m, n$  – целые числа и  $m > 0, n > 0$ . Разработать рекурсивную функцию, которая вычисляет площадь треугольника на основе зависимости:

$$S(n, m) = \begin{cases} 1, & \text{если } n = m = 1; \\ S(n-1, m) + m, & \text{если } n > 1; \\ S(n, m-1) + 1, & \text{если } m > 1. \end{cases}$$

12. Рассчитать значение квадрата целого положительного числа  $n$ , если известна зависимость  $n^2 = (n-1)^2 + 2 \cdot (n-1) + 1$   
Из этой зависимости получается формула рекурсии:

$$f(n) = \begin{cases} 1, & \text{если } n = 1; \\ f(n-1) + 2 \cdot n - 1, & \text{если } n > 1. \end{cases}$$

13. Вычислить количество комбинаций из  $n$  разных элементов по  $m$ .  
Количество комбинаций определяется формулой

$$C_n^m = \begin{cases} 1, & \text{если } m = 0, n > 0 \text{ або } m = n \geq 0; \\ 0, & \text{если } m > n \geq 0; \\ C_{n-1}^{m-1} + C_{n-1}^m & \text{иначе.} \end{cases}$$

14. Разработать рекурсивную функцию, которая реализует синтаксический анализатор для понятия идентификатор. Как известно, в языке программирования C# идентификатор подчиняется следующим правилам:
  - первым символом идентификатора обязательно есть буква латинского алфавита;

- второй, третий и т.д. символ идентификатора может быть буквой или цифрой.

Общая формула идентификатора следующая:

$$identifier ::= \left\{ \begin{array}{l} letter \\ identifier \left\{ \begin{array}{l} digit \\ letter \end{array} \right\} \end{array} \right\}$$

15. Вычислить значение функции Аккермана для двух неотрицательных целых чисел  $n$  и  $m$ , где

$$A(n, m) = \begin{cases} m+1, & \text{если } n = 0; \\ A(n-1, 1), & \text{если } n \neq 0, m = 0; \\ A(n-1, A(n, m-1)), & \text{если } n > 0, m > 0. \end{cases}$$

16. Составить программу, которая реализует синтаксический анализатор для понятия «expression»

$$\begin{aligned} Expression &::= \left\{ \begin{array}{l} simple\_identifier \\ \{ simple\_identifier \ operation \ simple\_identifier \} \end{array} \right\} \\ simple\_identifier &::= letter \\ operation &::= \left\{ \begin{array}{l} - \\ + \\ * \end{array} \right\}. \end{aligned}$$

где

*Expression* – некоторое простое выражение;

*simple\_identifier* – некоторый простой идентификатор;

*letter* – буква латинского алфавита от ‘a’ до ‘z’ или от ‘A’ до ‘Z’;

*operation* – знак операции ‘+’, ‘−’ или ‘\*’.

- 17 Составить программу, которая реализует синтаксический анализатор для понятия «сумма»

$$\begin{aligned} \text{Sum} &::= \text{Integer} \{ \text{Operation Integer} \} \\ \text{Integer} &::= \text{Digit} \{ \text{Digit} \} \\ \text{Operation} &::= \begin{cases} + \\ - \end{cases}. \end{aligned}$$

где

- *Sum* – функция, которая анализирует общую сумму;
- *Integer* – функция, которая анализирует целое число;
- *Digit* – функция, которая анализирует цифру;
- *Operation* – функция, которая реализует знак операции. Знаком операции может быть ‘+’ или ‘-’.

Выражение, взятое в фигурные скобки { } может повторяться за исключением операции.

18. Разработать рекурсивную функцию Min() поиска минимального значения в массиве A, который содержит  $n$  чисел типа double ( $n > 0$ ).
19. Рекурсивная функция ConvertAnyR(). Перевод числа из одной системы исчисления в другую
20. Рекурсивная функция isPrime(). Определить, есть ли число простым
21. Напишите программу на C для печати четных или нечетных чисел в заданном диапазоне с использованием рекурсии.
22. Напишите программу на C # Sharp, чтобы проверить, является ли данная строка палиндромной или не использует рекурсию.
23. Напишите программу на C # Sharp, чтобы найти факториал заданного числа с помощью рекурсии.
24. Напишите программу на C # Sharp, чтобы найти числа Фибоначчи для чисел ряда, используя рекурсию.
25. Напишите программу на C # Sharp для генерации всех возможных перестановок массива с использованием рекурсии.
26. Напишите программу на C # Sharp, чтобы найти LCM и GCD из двух чисел, используя рекурсию.

Тестовые данные :

Введите первое число: 10

Введите второе число: 15

Ожидаемый результат :

ГКД 10 и 15 = 5

LCM 10 и 15 = 30



27. Напишите программу на C # Sharp, чтобы получить обратную строку, используя рекурсию.

28. Напишите программу на C # Sharp для вычисления мощности любого числа с помощью рекурсии.

Тестовые данные :

Введите базовое значение: 5

Введите показатель степени: 3

*Ожидаемый результат :*

Значение 5 в степени 3 составляет: 125

29. Написать рекурсивный алгоритм определения квадрата числа

30. Написать рекурсивный алгоритм определения куба числа.

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## **Практическое занятие № 18**

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Создание диаграммы прецедентов и диаграммы классов

**Цель:** научиться создавать диаграммы прецедентов и диаграммы классов

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Индивидуальные задания:** Создать диаграмму прецедентов и диаграмму классов на основании предметной области. Варианты предметных областей представлены в таблице.

№	Предметная область
---	--------------------

варианта	
1	Автомастерская
2	Груминг салон
3	Модельное агентство
4	Телефонный автомат
5	Заказ такси
6	Библиотека
7	Налоговая служба
8	Заказ товаров через интернет
9	Почта
10	Учебная часть
11	Ресторан
12	Издательство книг
13	Стоматология
14	Салон красоты
15	Ателье индивидуального пошива одежды
16	Оптовый склад
17	Продажа подержанных автомобилей
18	Список фермерских хозяйств
19	Перевозки внутригородских маршрутов
20	Междугородные перевозки
21	Агентство по продаже авиабилетов
22	Гостиница
23	Выставка животных
24	Кадровое агентство
25	Туристическое агентство
26	Справочник меломана
27	Ежедневник
28	Ломбард
29	Банковский справочник
30	Агентство недвижимости

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки

в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## **Практическое занятие № 19**

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Создание диаграммы последовательностей и диаграммы состояний

**Цель:** научиться создавать диаграммы последовательностей и состояний

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Индивидуальные задания:** Создать диаграмму последовательностей и диаграмму на основании предметной области. Варианты предметных областей взять из предыдущего задания.

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 20

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Работа с классами

**Цель:** разработать класс, содержащий конструктор и методы

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

**Последовательность выполнения работы**

1. Запустить MS Visual Studio
2. Создать консольное приложение
3. Выполнить задание (по вариантам)
4. Сохранить файл

**Индивидуальные задания :**

Для всех вариантов задач создать класс с указанными двумя полями (Поле 1, Поле 2) и тремя методами:

- конструктор для инициализации объекта;
- функция формирования строки с информацией об объекте;
- функция обработки значений полей по индивидуальному варианту

№ вар.	Поле 1	Поле 2	Функция обработки полей
1	Номинал купюры (1, 2, 5, 10 и т.д.)	Количество купюр	Вычислить сумму купюр
2	Номинал монеты (1, 2, 5, 10 и т.д.)	Количество монет	Вычислить сумму монет
3	Цена товара	Количество единиц товара	Вычислить общую стоимость товара
4	Калорийность 100г продукта	Вес продукта в граммах	Вычислить общую калорийность продукта
5	Вещественное число – левая граница диапазона	Вещественное число – правая граница диапазона	Квадрат длины диапазона
6	Количество минут	Количество секунд	Вычислить общее количество секунд
7	Количество часов	Количество минут	Вычислить общее количество минут

8	Вещественное число – первый катет прямоугольного треугольника	Вещественное число – второй катет прямоугольного треугольника	Вычислить площадь прямоугольного треугольника
9	Вещественное число – скорость движения (м/сек)	Целое число – время движения в минутах	Вычислить пройденное расстояние (в метрах)
10	Вещественное число – первый катет прямоугольного треугольника	Вещественное число – второй катет прямоугольного треугольника	Вычислить длину гипотенузы прямоугольного треугольника
11	Целое число – нижнее основание трапеции	Целое число – верхнее основание трапеции	Вычислить полу-сумму оснований трапеции
12	Вещественное число – первый катет прямоугольного треугольника	Вещественное число – второй катет прямоугольного треугольника	Вычислить тангенс угла $\alpha$ , противолежащего второму катету прямоугольного треугольника
13	Вещественное число	Вещественное число	Вычислить полу-разность чисел
14	Вещественное число	Вещественное число	Вычислить корень квадратный из произведения чисел
15	Целое число – $x$	Целое число – $y$	Вычислить целую часть от деления $x$ на $y$
16	Целое число – $x$	Целое число – $y$	Вычислить квадрат меньшего из чисел
17	Целое число – $x$	Целое число – $y$	Вычислить куб большего из чисел
18	Продолжительность телефонного разговора в минутах	Стоимость одной минуты разговора	Вычислить общую стоимость разговора
19	Координата точки на плоскости (по горизонтали)	Координата точки на плоскости (по вертикали)	Определить периметр прямоугольника, ограниченного координатами точки и осями $Ox$ и $Oy$
20	Вещественное число – $a$	Вещественное число – $b$	Вычислить разность квадратов чисел $a^2 - b^2$
21	Вещественное число – $a$	Вещественное число – $b$	Вычислить сумму квадратов чисел $a^2 + b^2$

22	Координата точки на плоскости (по горизонтали) – $x_1$	Координата точки на плоскости (по вертикали) – $y_1$	Определить площадь прямоугольника, ограниченного координатами точки и осями $Ox$ и $Oy$
23	Координата точки на плоскости (по горизонтали) – $x_1$	Координата точки на плоскости (по вертикали) – $y_1$	Вычислить расстояние от точки до начала координат
24	Количество часов работы	Тариф оплаты за час работы	Общая стоимость работы
25	Радиус окружности	Угол в радианах	Вычислить длину дуги
26	Радиус окружности основания	Высота цилиндра	Вычислить площадь поверхности цилиндра
27	Радиус окружности основания конуса	Высота конуса	Вычислить объем конуса
28	Напряжение (в Вольтах)	Сопротивление (в Омах)	Вычислить значение тока (в Амперах)
29	Ток в амперах	Сопротивление резистора $R_1$ (в Омах)	Вычислить мощность на участке электрической цепи (в Ваттах)
30	Масса тела – $m$ (в граммах)	Скорость движения – $v$ (в м/с)	Вычислить кинетическую энергию движущегося тела $W_k = \frac{mv^2}{2}$

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 21

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Перегрузка методов

**Цель:** выполнить перегрузку методов класса

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

### Последовательность выполнения работы

1. Запустить MS Visual Studio
2. Создать консольное приложение
3. Выполнить задание
4. Сохранить файл

### Индивидуальные задания:

1. Создать класс с полями, указанными в индивидуальном задании (столб 2).
2. Реализовать в классе методы:
  - конструктор по умолчанию;
  - конструктор перезагрузки с параметрами;
  - деструктор для освобождения памяти (с сообщением об уничтожении объекта);
  - функции обработки данных (1 и 2), указанные в индивидуальном задании (столбцы 3 и 4);
  - функцию формирования строки информации об объекте.

№ вар.	Класс-родитель и его поля	Функция-метод 1 обработки данных	Функция-метод 2 обработки данных
1	Дата (три числа): день, месяц, год	Определить, является ли год високосным (кратным 4)	Увеличить дату на 5 дней
2	Дата (три числа): день, месяц, год	Увеличить год на 1	Уменьшить дату на 2 дня
3	Дата (три числа): день, месяц, год	Определить, совпадают ли номер месяца и число дня	Увеличить дату на один месяц
4	Время (три числа): часы, минуты, секунды	Вычислить количество секунд в указанном времени	Увеличить время на 5 секунд



5	Время (три числа): часы, минуты, секунды	Вычислить количество полных минут в указанном времени	Уменьшить время на 10 минут
6	Время (три числа): часы, минуты, секунды	Определить количество минут до полуночи (24:00:00)	Увеличить время 100 минут
7	Координаты изображения прямоугольника: $x1, y1, x2, y2$	Вычислить площадь прямоугольника в пикселях	Изобразить прямоугольник на форме (Image) с толщиной линии 2 пикселя
8	Координаты изображения прямоугольника: $x1, y1, x2, y2$	Вычислить длину диагонали прямоугольника в пикселях	Изобразить прямоугольник и его диагональ на форме (Image)
9	Координаты изображения прямоугольника: $x1, y1, x2, y2$	Определить, является ли прямоугольник квадратом?	Изобразить прямоугольник на форме (Image), закрашенный зеленым цветом
10	Правильная дробь: числитель, знаменатель	Выразить значение дроби в процентах	Найти сумму цифр значения знаменателя
11	Комплексное число: действительная ( $a1$ ) и мнимая ( $b1$ ) части числа	Вычислить модуль комплексного числа	Найти комплексное число, обратное заданному
12	Комплексное число: действительная и мнимая часть числа	Вычислить произведение комплексного числа на число, вводимое пользователем	Вычислить аргумент комплексного числа в градусах
13	Книга: название, количество страниц, цена	Вычислить среднюю стоимость одной страницы	Увеличить цену книги в два раза, если название начинается со слова «Программирование»
14	Книга: название, автор, год издания	Вычислить, сколько лет книге	Количество дней, прошедших после года издания книги
15	Работник: фамилия, оклад, год поступления на работу	Вычислить стаж работы работника на данном предприятии	Сколько дней прошло после года поступления на работу

16	Работник: фамилия, оклад, год рождения	Вычислить возраст работника	Сколько календарных дней до исполнения работнику 50 лет
17	Вектор на плоскости: координаты вектора на плоскости ( $x_1, y_1, x_2, y_2$ )	Вычислить длину вектора	Изобразить линию вектора на форме (Image) с толщиной линии 2 пикселя
18	Вектор на плоскости: координаты вектора на плоскости ( $x_1, y_1, x_2, y_2$ )	Вычислить координаты середины вектора	Равен ли угол наклона вектора 45 градусов?
19	Вектор на плоскости: координаты вектора на плоскости ( $x_1, y_1, x_2, y_2$ )	Вычислить координаты вектора, удвоенной длины	Вычислить площадь прямоугольного треугольника, образованного вектором и прямыми, параллельными осям $Ox, Oy$ .
20	Цилиндр: диаметр основания, высота	Вычислить объем цилиндра	Изобразить круг заданного диаметра на форме (Image), закрашенный красным цветом.
21	Параллелепипед: длины сторон	Вычислить объем параллелепипеда	Вычислить длину наибольшей диагонали параллелепипеда
22	Параллелепипед: длины сторон	Вычислить площадь поверхности	Вычислить сумму длин всех ребер параллелепипеда.
23	Четыре целых числа: $a, b, c, d$	Вычислить среднее арифметическое чисел	Определить максимальное из чисел
24	Три вещественных числа $x, y, z$	Вычислить среднее геометрическое чисел	Определите, сколько цифр содержит сумма заданных трех чисел.
25	Товар: наименование, цена, год выпуска	Определить, сколько лет назад был выпущен товар	Увеличить цену товара на 20%, если в наименовании товара есть слово «TV».
26	Товар: наименование, цена в гривне, изготовитель	Пересчитать цену товара в долларах	Увеличить цену товара в долларах, если название товара содержит слово «Toyota».

27	Координаты изображения эллипса: $x_1, y_1, x_2, y_2$	Определить, является ли эллипс окружностью?	Изобразить эллипс на форме (Image) зеленым цветом .
28	Книга: название, количество страниц, цена	Увеличить количество страниц на 10	Уменьшить цену в два раза, если количество страниц больше 100 (после увеличения)
29	Комната: длина, ширина, высота (в метрах)	Площадь стен (вместе с окнами и дверьми)	Площадь стен без окна (размер $2 \times 15$ м) и двери (размер $2 \times 8$ м).
30	Работник: фамилия, должность, оклад	Увеличить оклад на 15% (каждому работнику)	Работникам, у которых фамилия начинается с сочетания букв «Иван», присвоить должность «инженер».

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### Практическое занятие № 22

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Определение операций в классе

**Цель:** написать класс, определить операции в классе

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

## Последовательность выполнения работы

1. Запустить MS Visual Studio
2. Создать консольное приложение
3. Выполнить задание (по вариантам)
4. Сохранить файл

## Индивидуальные задания:

Для класса, созданного в задании 18 создать класс-потомок с дополнительным полем, указанным в индивидуальном задании (столб 2).

Реализовать в классе-потомке методы:

- конструктор;
- функцию обработки данных, указанную в индивидуальном задании (столб 3).

№ вар.	Поле класса-потомка	Функция обработки данных
1	Стоимость одного евро (€) в гривне	Стоимость купюр в евро
2	Стоимость одного \$ (доллара) в лире	Стоимость монет в центах
3	Год выпуска товара	Сколько лет товару
4	Количество витамина С в 1 грамме продукта	Количество витамина С в продукте
5	Вещественное число $x$	Проверить, принадлежит ли число $x$ заданному диапазону
6	Скорость движения объекта наблюдения (в м/сек)	Расстояние, пройденное объектом наблюдения
7	Длительность выполнения одной операции в минутах	Сколько операций можно выполнить за указанное время
8	Высота призмы	Объем призмы, у которой в основании прямоугольный треугольник
9	Сила, приложенная к движущемуся объекту	Количество работы, выполненной при прямолинейном перемещении объекта
10	Высота призмы	Сумму всех ребер призмы, у которой в основании прямоугольный треугольник
11	Высота трапеции	Площадь трапеции
12	Значение угла $\beta$ в радианах	Разность между заданным углом $\beta$ и углом $\alpha$ в прямоугольном треугольнике
13	Вещественное число $c$	Определить произведение полуразности чисел класса-родителя ( $a$ и $b$ ) на число $c$

14	Вещественное число – $z$	Вычислить выражение $xy + z$ , где $x$ и $y$ – поля класса-родителя
15	Вещественное число – $z$	Вычислить выражение $\frac{x + y}{z}$ , где $x$ и $y$ – поля класса-родителя
16	Вещественное число – $z$	Произведение числа $z$ на минимальное из чисел $x$ и $y$ (поля класса-родителя)
17	Вещественное число – $z$	Сумма куба числа $z$ и максимального из чисел $x$ и $y$ (поля класса-родителя)
18	Количество разговоров по телефону за сутки	Общая стоимость разговоров за сутки
19	Вещественное число – $c$	Увеличить обе координаты точки на $c$ и найти их произведение
20	Вещественное число – $x$	Вычислить для заданного числа $x$ значение выражения $a x^2 + b$
21	Вещественное число – $c$	Вычислить для заданного числа $c$ корень уравнения $a x + b = c$
22	Координаты второй точки на плоскости: $x_2, y_2$	Найти расстояние между первой и второй точкой
23	Радиус окружности	Определить, находится ли точка с параметрами $x_1, y_1$ (класса-родителя) внутри окружности с центром в начале координат
24	Число – подоходный налог в процентах	Вычислить, сколько денег получит работник, если вычтут подоходный налог
25	Число – высота объемного сектора	Вычислить объем фигуры, у которой в основании сектор окружности с параметрами класса-родителя
26	Количество одинаковых цилиндров	Общая площадь поверхностей цилиндров
27	Высота отпиленной сверху части конуса	Объем усеченной пирамиды, оставшейся после отпиливания
28	Время в секундах	Работа, выполненная резистором за указанное время
29	Сопротивление второго, последовательно соединенного резистора $R_2$	Определить общую мощность на двух резисторах
30	Высота расположения тела движущегося тела	Определить потенциальную энергию тела



### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 23**

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Создание наследованных классов

**Цель:** Создание простой иерархии классов и изучение методов инициализации объектов производных классов.

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

**Последовательность выполнения работы**

1. Запустить MS Visual Studio
2. Создать консольное приложение
3. Выполнить задание (по вариантам)
4. Сохранить файл

**Индивидуальные задания:**

Для класса, созданного в задании 19 создать класс-потомок с полями, указанными в индивидуальном задании (столб 2).

Реализовать в классе-потомке методы:

- конструктор;
- функцию обработки данных, указанную в индивидуальном задании (табл. столб 3);
- функцию формирования строки информации об объекте.

Создать проект для демонстрации работы: ввод и вывод информации об объектах: классе-родителе и классе-потомке.

№ вар.	Класс-родитель и его поля (из табл. 11.2)	Класс-потомок и его поля (поля класса- родителя выделены курсивом)	Функция-метод обработ- ки данных объекта класса-потомка
1	Дата (три числа): день, месяц, год	Список друзей: ФИО, телефон, <i>дата</i> рож- дения,	Количество дней до дня очередного рождения
2	Дата (три числа): день, месяц, год	Работник: ФИО, <i>дата</i> поступления на предприятие	Количество лет работы на предприятии
3	Дата (три числа): день, месяц, год	Лекарство: наимено- вание, <i>дата</i> выпуска, фирма	Сколько прошло дней от изготовления лекарства
4	Время (три числа): часы, минуты, секунды	Расписание движения поездов: номер поезда, направление, <i>время</i> отправления	Количество минут до отправления поезда с указанным номером и введенное время
5	Время (три числа): часы, минуты, секунды	Абонент мобильной связи: фамилия, оператор, текущее <i>время</i>	Определить, является ли время льготным для абонента (время от 0 до 8 часов)
6	Координаты изображения прямоугольника: $x1, y1, x2, y2$	Изображение конвер- та (прямоугольник с линиями диагоналей): <i>координаты прямо- угольника</i> , цвет линий	Площадь верхнего (над- диагонального) треуголь- ника в пикселях
7	Время (три числа): часы, минуты, секунды	Расписание занятий: дисциплина, <i>время</i> начала, аудитория	Какая дисциплина по расписанию начинается в указанное время
8	Координаты изображения прямоугольника: $x1, y1, x2, y2$	Изображение прямо- угольника с вписан- ным в его центр кругом: <i>координаты прямоугольника</i> , радиус круга $R$ ( $R < x2$ $- x1, R < y2 - y1$ )	Площадь фигуры между прямоугольником и кругом



9	Координаты изображения прямоугольника: $x_1, y_1, x_2, y_2$	Изображение закрашенного прямоугольника с текстом: <i>координаты прямоугольника</i> , заданный текст, цвет закрашивания	Произведение периметра и длины диагонали прямоугольника в пикселях
10	Правильная дробь: числитель, знаменатель	Смешанная дробь: целая часть, <i>числитель и знаменатель</i>	Представить смешанную дробь в виде десятичного вещественного числа.
11	Комплексное число: действительная ( $a_1$ ) и мнимая ( $b_1$ ) части числа	Два комплексных числа: <i>действительная (<math>a_1</math>) и мнимая (<math>b_1</math>) части первого числа</i> ; действительная ( $a_2$ ) и мнимая ( $b_2$ ) части второго числа	Вычислить произведение двух комплексных чисел.
12	Комплексное число: действительная и мнимая часть числа	<i>Комплексное</i> сопротивление: сопротивление резистора ( <i>действительная часть</i> ), значение индуктивности ( <i>мнимая часть</i> ), угловая частота	Вычислить модуль и аргумент комплексной проводимости участка цепи «резистор - индуктивность».
13	Книга: название, количество страниц, цена	Библиотека: <i>название, количество страниц, цена</i> , скидка в процентах	Стоимость книги с учетом скидки.
14	Книга: название, автор, год издания	Книжный магазин: <i>название, автор, год издания</i> , цена	Уменьшить стоимость книги на 20%, если книге больше 5 лет.
15	Работник: фамилия, оклад, год поступления на работу	Работники предприятия: <i>фамилия, оклад, год поступления на работу</i> , год рождения	Определить, сколько лет нужно работать работнику до 60 лет, а если ему больше 60, то сколько лет он работает после 60 лет.
16	Работник: фамилия, оклад, год рождения	Работники фирмы: <i>фамилия, оклад, год рождения</i> , должность	Увеличить оклад работникам с должностью программист на 20%.

17	Вектор на плоскости: координаты вектора на плоскости ( $x_1, y_1, x_2, y_2$ )	Вектор и точка на плоскости: <i>координаты вектора</i> ( $x_1, y_1, x_2, y_2$ ); координаты точки— $x_3, y_3$	Определить площадь треугольника, образованного вектором и точкой.
18	Вектор на плоскости: координаты вектора на плоскости ( $x_1, y_1, x_2, y_2$ )	Два вектора с общим началом ( $x_1, y_1$ ) на плоскости: <i>координаты первого вектора</i> — $x_1, y_1, x_2, y_2$ ; координаты второго вектора — $x_1, y_1, x_3, y_3$	Определить координаты вектора суммы двух векторов.
19	Вектор на плоскости: координаты вектора на плоскости ( $x_1, y_1, x_2, y_2$ )	Два параллельных вектора на плоскости одинаковой длины: <i>координаты первого вектора</i> — $x_1, y_1, x_2, y_2$ ; второй вектор смещен вправо по оси $Ox$ на величину $a$ , второй — вниз по оси $Oy$ на $b$	Определить площадь параллелограмма, образованного этими векторами и линиями, соединяющих их начала и концы.
20	Цилиндр: диаметр основания, высота	Изолированный провод: <i>диаметр, длина, удельный вес; толщина изоляции и её удельный вес</i>	Определить вес изолированного провода.
21	Параллелепипед: длины сторон	Металлический брус: <i>ширина, высота, длина, удельный вес</i>	Определить вес металлического бруса.
22	Параллелепипед: длины сторон	Балка с прямоугольным сечением: <i>ширина, высота, длина, удельный вес; количество равных частей, на которое её распилят</i>	Площадь поверхности одной части распиленной балки и её вес.
23	Четыре целых числа: $a, b, c, d$	Пять чисел: <i>четыре целых числа</i> ( $a, d, c, d$ ) и число $x$	Вычислить сумму квадратов разности каждого из четырех чисел и числа $x$ .

24	Три вещественных числа: $x, y, z$	Два набора чисел: <i>три вещественных числа <math>x, y, z</math> и три вещественных числа <math>a, b, c</math></i>	Определить скалярное произведение двух наборов чисел.
25	Товар: наименование, цена, год выпуска	Фирменный товар: <i>наименование, цена, год выпуска, дата поступления товара</i>	Количество дней после года выпуска товара до текущего дня.
26	Товар: наименование, цена в гривне, изготовитель	Товар: <i>наименование, цена в гривне, изготовитель, год выпуска, скидка в процентах</i>	Изменить стоимость товара с учетом скидки для товаров, изготовленных фирмой более двух лет назад.
27	Координаты изображения эллипса: $x_1, y_1, x_2, y_2$	Дуга эллипса: <i>координаты изображения эллипса <math>x_1, y_1, x_2, y_2</math>, координаты концов дуги <math>x_3, y_3, x_4, y_4</math></i>	Построить изображение дуги эллипса на форме (Image) синим цветом, толщиной линии 2 пикселя.
28	Книга: название, количество страниц, цена	Изданная книга: <i>название, количество страниц, цена, автор книги, дата поступления в типографию</i>	Сколько дней книга находилась в типографии.
29	Комната: длина, ширина, высота (в метрах)	Помещения для офисов: <i>длина, ширина, высота комнат, количество комнат и площадь коридора, расход краски на <math>1 \text{ м}^2</math></i>	Определить количество краски, необходимое для покраски стен и потолка помещений офиса (в каждой комнате одно окно размером $2 \times 15 \text{ м}$ ).
30	Работник: фамилия, должность, оклад	Работники предприятия: <i>фамилия, должность, оклад, рейтинг (в 100-бальной системе)</i>	Увеличить оклад работников на 20%, если их рейтинг от 60 до 75 баллов, на 40%, если их рейтинг от 75 до 90 баллов, на 60%, если их рейтинг от 90 до 100 баллов.

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 24**

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Полиморфизм классов

**Цель:** Изучение полиморфизма классов

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** файл с программным кодом

**Последовательность выполнения работы**

1. Запустить MS Visual Studio
2. Создать консольное приложение
3. Выполнить задание (по вариантам)
4. Сохранить файл

**Индивидуальные задания:**

построить класс 1-го уровня с указанными в индивидуальном задании полями и методами:

- конструктор;
- функция, которая определяет «качество» объекта –  $Q$  по заданной формуле (столб 2);
- вывод информации об объекте.

Построить класс 2-го уровня (класс-потомок), который содержит:

- дополнительное поле  $P$ ;
- функция, которая определяет «качество» объекта класса 2-го уровня –  $Q_p$ , которая перекрывает функцию качества класса 1-го уровня ( $Q$ ), выполняя вычисление по новой формуле (табл. столб 3).

Создать проект для демонстрации работы: ввод и вывод информации об объектах классов 1-го и 2-го уровней.

№ вар	Поля и функция «качества» ( $Q$ ) класса 1-го уровня	Поле и функция «качества» $Q_p$ класса 2-го уровня
----------	---------------------------------------------------------	-------------------------------------------------------

1	<p>Компьютер:</p> <ul style="list-style-type: none"> <li>- наименование процессора;</li> <li>- тактовая частота процессора (МГц);</li> <li>- объем оперативной памяти (Мб).</li> </ul> <p><math>Q = (0,1 \cdot \text{частота}) + \text{память}</math></p>	<p><math>P</math>: объем винчестера (Гб)</p> <p><math>Qp = Q + 0,5 \cdot P</math></p>
2	<p>Оператор мобильной связи:</p> <ul style="list-style-type: none"> <li>- название оператора;</li> <li>- стоимость 1 минуты разговора;</li> <li>- площадь покрытия.</li> </ul> <p><math>Q = 100 \cdot \text{площадь покрытия} / \text{стоимость 1 минуты разговора}</math></p>	<p><math>P</math>: наличие платы за каждое соединение</p> <p><math>Qp = 0,7 \cdot Q</math>, если <math>P</math> - истина, иначе <math>Qp = 1,5 \cdot Q</math></p>
3	<p>Товар на складе:</p> <ul style="list-style-type: none"> <li>- наименование;</li> <li>- цена;</li> <li>- количество.</li> </ul> <p><math>Q = \text{цена} / \text{количество}</math></p>	<p><math>P</math>: год выпуска товара</p> <p><math>Qp = Q + 0,5 \cdot (T - P)</math>, где <math>T</math> - текущий год</p>
4	<p>Кабель:</p> <ul style="list-style-type: none"> <li>- тип;</li> <li>- количество жил кабеля;</li> <li>- диаметр.</li> </ul> <p><math>Q = \text{диаметр} / \text{количество жил}</math></p>	<p><math>P</math>: наличие оплетки</p> <p><math>Qp</math>: если <math>P</math> - истина, то <math>Qp = 2 \cdot Q</math>; иначе <math>Qp = 0,7 \cdot Q</math></p>
5	<p>Учебник по программированию:</p> <ul style="list-style-type: none"> <li>- название;</li> <li>- количество страниц;</li> <li>- цена.</li> </ul> <p><math>Q = \text{цена} / \text{количество страниц}</math></p>	<p><math>P</math>: год издания</p> <p><math>Qp = Q - 0,2 \cdot (T - P)</math>, где <math>T</math> - текущий год</p>
6	<p>Мобильный телефон:</p> <ul style="list-style-type: none"> <li>- марка;</li> <li>- цена;</li> <li>- объем памяти.</li> </ul> <p><math>Q = \text{объем памяти} / \text{цена}</math></p>	<p><math>P</math>: количество SIM карт</p> <p><math>Qp = Q \cdot P</math></p>
7	<p>Спутниковая антенна (тарелка):</p> <ul style="list-style-type: none"> <li>- диаметр;</li> <li>- материал;</li> <li>- цена.</li> </ul> <p><math>Q = \text{диаметр} / \text{цена}</math></p>	<p><math>P</math>: тип подвески (азимутальная, полярная, тороидальная)</p> <p><math>Qp = Q</math>, если тип подвески азимутальный, <math>Qp = 2 \cdot Q</math>, если тип подвески полярный, <math>Qp = 2,5 \cdot Q</math>, если тип подвески тороидальный</p>

8	<p>Экзамен:</p> <ul style="list-style-type: none"> <li>- дисциплина;</li> <li>- число студентов на экзамене;</li> <li>- продолжительность экзамена (ч).</li> </ul> <p><math>Q</math> = число студентов / продолжительность</p>	<p><math>P</math>: процент двоек</p> <p><math>Qp = Q \cdot (100 - P) / 100</math></p>
9	<p>Спортсмен:</p> <ul style="list-style-type: none"> <li>- фамилия;</li> <li>- число соревнований;</li> <li>- сумма мест, занятых спортсменом в соревнованиях.</li> </ul> <p><math>Q</math> = (число соревнований) / (сумма мест)</p>	<p><math>P</math>: занимал ли хотя бы раз первое место</p> <p><math>Qp = 1,5 \cdot Q</math>, если <math>P</math> - истина, иначе – <math>Qp = Q</math>,</p>
10	<p>Программист:</p> <ul style="list-style-type: none"> <li>- фамилия;</li> <li>- число программ, написанных программистом;</li> <li>- число языков программирования, которыми он пишет программы.</li> </ul> <p><math>Q</math> = (число программ) * (число языков)</p>	<p><math>P</math>: число программ, которые работают правильно</p> <p><math>Qp = Q \cdot P /</math> (число всех программ)</p>
11	<p>Спектакль:</p> <ul style="list-style-type: none"> <li>- название;</li> <li><math>n1</math> – число зрителей в начале;</li> <li><math>n2</math> – число зрителей в конце.</li> </ul> <p><math>Q = (n2 - n1) / n1</math></p>	<p><math>P</math>: год написания пьесы</p> <p><math>Qp = Q \cdot (T - P + 1)</math>, где <math>T</math> - текущий год</p>
12	<p>Алмаз:</p> <ul style="list-style-type: none"> <li>- название;</li> <li>- вес (в каратах);</li> <li>- качество огранки в баллах (число).</li> </ul> <p><math>Q = 0,4 \cdot \text{вес} + 0,6 \cdot \text{качество огранки}</math></p>	<p><math>P</math>: цвет (белый, голубой, желтый, и тп)</p> <p><math>Qp</math>: если цвет голубой, то <math>Qp = Q + 1</math>;</p> <p>а если желтый, то <math>Qp = Q - 0,5</math></p> <p>иначе <math>Qp = Q</math></p>
13	<p>Компьютерная сеть:</p> <ul style="list-style-type: none"> <li>- название организации;</li> <li>- число рабочих станций;</li> <li>- среднее расстояние между станциями (м).</li> </ul> <p><math>Q</math> = число станций · среднее расстояние</p>	<p><math>P</math>: средняя скорость передачи данных в сети (Мб/с)</p> <p><math>Qp = Q \cdot P</math></p>

14	<p>Армия:</p> <ul style="list-style-type: none"> <li>- вид войск;</li> <li>- численность (тыс человек);</li> <li>- вооруженность (баллы - число).</li> </ul> <p><math>Q=0,3 \cdot \text{численность} + 0,7 \cdot \text{вооруженность}</math></p>	<p><math>P</math>: опыт (число месяцев, на протяжении которых армия вела боевые действия)</p> <p><math>Qp = Q \cdot (P+1)</math></p>
15	<p>Автомобиль:</p> <ul style="list-style-type: none"> <li>- марка автомобиля;</li> <li>- мощность двигателя (кВт);</li> <li>- число мест.</li> </ul> <p><math>Q = 0,1 \cdot \text{мощность} \cdot \text{число мест}</math></p>	<p><math>P</math>: год изготовления</p> <p><math>Qp = Q - 1,5 \cdot (T - P)</math>, где <math>T</math> - текущий год</p>
16	<p>Партия:</p> <ul style="list-style-type: none"> <li>- название;</li> <li>- численность (тыс. членов);</li> <li>- процент голосов на последних выборах.</li> </ul> <p><math>Q = 0,3 \cdot \text{численность} + 0,7 \cdot \text{процент гол}</math></p>	<p><math>P</math>: численность партии в прошлом году</p> <p><math>Qp</math>: если численность в текущем году увеличилась, то <math>Qp = 1,2 \cdot Q</math>; а если сократилась, то <math>Qp = 0,8 \cdot Q</math></p>
17	<p>Высшее учебное заведение:</p> <ul style="list-style-type: none"> <li>- название заведения;</li> <li>- количество студентов, зачисленных на 1-й курс;</li> <li>- количество выпускников.</li> </ul> <p><math>Q = \text{количество выпускников} / \text{количество зачисленных}</math></p>	<p><math>P</math>: процент выпускников, которые работают по специальности</p> <p><math>Qp = P \cdot Q</math></p>
18	<p>Солдат:</p> <ul style="list-style-type: none"> <li>- фамилия;</li> <li>- рост (м);</li> <li>- вес (кг).</li> </ul> <p><math>Q = \text{рост} \cdot \text{вес}</math></p>	<p><math>P</math>: образование (начальное, среднее, высшее)</p> <p><math>Qp</math>: если образование высшее, то <math>Qp = 2 \cdot Q</math>; а если начальное, то <math>Qp = 0,5 \cdot Q</math>; иначе <math>Qp = Q</math></p>
19	<p>Телевизор:</p> <ul style="list-style-type: none"> <li>- фирма;</li> <li>- диагональ экрана (дюйм);</li> <li>- звуковая мощность (дб).</li> </ul> <p><math>Q = \text{диагональ} + (0,05 \cdot \text{мощность})</math></p>	<p><math>P</math>: страна-производитель</p> <p><math>Qp</math>: если страна - Япония, то <math>Qp = 2 \cdot Q</math>; а если Сингапур или Корея, то <math>Qp = 1,5 \cdot Q</math>; иначе <math>Qp = Q</math></p>
20	<p>Митинг:</p> <ul style="list-style-type: none"> <li>- название события;</li> </ul> <p><math>n1</math> – число ораторов; <math>n2</math> – число участников.</p> <p><math>Q = n1/n2</math></p>	<p><math>P</math>: число групп ораторов, которые высказывали одинаковые мысли</p> <p><math>Qp = Q + P/n2</math></p>

21	Дом: - номер дома; - число квартир; - год сооружения. $Q = (\text{число квартир}) + 2 \cdot (T - \text{год сооружения})$ , где $T$ - текущий год	$P$ : район (центр, окраина, и тп)  $Q_p$ : если район - центр, то $Q_p = 2 \cdot Q$ ; иначе $Q_p = 0,5 \cdot Q$
22	Руководитель: - фамилия; - самооценка (в баллах - целое число); - оценка другими людьми (в баллах). $Q = (\text{оценка другими}) / \text{самооценка}$	$P$ : оценка потомками (в баллах)  $Q_p = 0,3 \cdot Q + 0,7 \cdot P$
23	Студент: - фамилия; - средний балл; - курс.  $Q = 0,2 \cdot \text{средний балл} \cdot \text{курс}$	$P$ : изучает дисциплины на английском языке  $Q_p = 2 \cdot Q$ , если $P$ - истина, иначе $Q_p = 0,9 \cdot Q$
24	Антенна: - название; - мощность; - высота (м). $Q = \text{мощность} + 0,5 \cdot \text{высота}$	$P$ : коэффициент излучения  $Q_p = Q - 0,1 \cdot P$
25	Самолет: - марка; - количество двигателей; - высота полета.  $Q = \text{кол-во двигателей} \cdot \text{высота полета} / 1000$	$P$ : страна-производитель  $Q_p$ : если страна - Россия, то $Q_p = Q + 1$ ; а если Франция, то $Q_p = Q + 0,5$
26	Студент: - фамилия; - число экзаменов; - число оценок «пять».  $Q = \text{число оценок «пять»} / \text{число экзаменов}$	$P$ : число оценок «три»  $Q_p = Q - 0,5P$
27	Фирма: - название; - доход (тыс \$ ); - рейтинг (в баллах). $Q = \text{доход} \cdot \text{рейтинг}$	$P$ : инвестиции в фирму (тыс \$ )  $Q_p = P^3 + Q$



28	Военный корабль: - название; - длина; - число пушек главного калибра. $Q = (\text{число пушек}) / \text{длина}$	$P$ : крейсерская скорость (в морских узлах) $Qp = 0.25Q + P$
29	Коробка спичек: - фирма изготовитель; - число спичек в коробке; - время горения одной спички (с). $Q = (\text{число спичек}) \cdot \text{время}$	$P$ : средний % бракованных спичек в коробке $Qp = (100 - P)Q / 100$
30	Полководец: - фамилия; - число битв; - число побед. $Q = (\text{число побед})^2 / (\text{число битв})$	$P$ : число побед с меньши- ми, чем у противника, силами $Qp = P^2 / \text{битвы} + Q$

#### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## **Практическое занятие № 25-26**

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Работа с объектами через интерфейсы

**Цель:** Создать класс, реализовать в нем метод CompareTo

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** файл с программным кодом

### **Последовательность выполнения работы**

5. Запустить MS Visual Studio
6. Создать консольное приложение
7. Выполнить задание (по вариантам)
8. Сохранить файл

### **Индивидуальные задания:**

**Задание 1.** Создать класс Figure с методами вычисления площади и периметра, а также методом, выводящим информацию о фигуре на экран.

Создать производные классы: Rectangle (прямоугольник), Circle (круг), Triangle (треугольник) со своими методами вычисления площади и периметра.

В классе Figure реализовать метод CompareTo так, чтобы можно было отсортировать объекты по их площадям.

### **Задание 2**

Создать класс Персона с методами, позволяющим вывести на экран информацию о персоне, а также определить ее возраст (на момент текущей даты).

Создать производные классы: Абитуриент (фамилия, дата рождения, факультет), Студент (фамилия, дата рождения, факультет, курс), Преподаватель (фамилия, дата рождения, факультет, должность, стаж), со своими методами вывода информации на экран, и определения возраста.

В классе Персона реализовать метод CompareTo так, чтобы можно было отсортировать базу данных о персонах по дате их рождения.

### **Задание 3**

Создать класс Издание с методами, позволяющим вывести на экран информацию об издании, а также определить, является ли данное издание искомым.

Создать производные классы: Книга (название, фамилия автора, год издания, издательство), Статья (название, фамилия автора, название журнала, его номер и год издания), Электронный ресурс (название, фамилия автора, ссылка, аннотация) со своими методами вывода информации на экран.

В классе Издание реализовать метод CompareTo так, чтобы можно было отсортировать каталог изданий по фамилии автора.

#### **Задание 4**

Создать класс Транспорт с методами, позволяющими вывести на экран информацию о транспортном средстве, а также определить его грузоподъемность.

Создать производные классы: Легковая\_машина (марка, номер, скорость, грузоподъемность), Мотоцикл (марка, номер, скорость, грузоподъемность, наличие коляски; при этом, если коляска отсутствует, то грузоподъемность равна 0), Грузовик (марка, номер, скорость, грузоподъемность, наличие прицепа; при этом, если есть прицеп, то грузоподъемность увеличивается в два раза) со своими методами вывода информации на экран, и определения грузоподъемности.

В классе Транспорт реализовать метод CompareTo так, чтобы можно было отсортировать базу данных о машинах по их грузоподъемности.

#### **Задание 5.**

Создать класс Автомобиль со свойствами: Название, Максимальная скорость (в км/ч). Определить 2 виртуальных метода: метод «Стоимость» – стоимость автомобиля, рассчитываемую по формуле. Максимальная скорость \* 100 и метод «Обновление модели», увеличивающий максимальную скорость на 10. Определить также метод «Информация», который возвращает строку, содержащую информацию об объекте: Название, Максимальную скорость и Стоимость.

Создать также класс наследник Представительский автомобиль, в котором переопределить методы: метод «Стоимость» возвращает число, равное. Максимальная скорость \* 250, а метод «Обновление модели» увеличивает скорость на 5 км/ч.

В главной программе (либо по нажатию на кнопку) создать объект класса Автомобиль с максимальной скоростью 140 км/ч и класса Представительский автомобиль с максимальной скоростью 160 км/ч. Вывести на экран (или форму) информацию об автомобилях. Обновить модели автомобилей и снова вывести информацию о них.

### Задание 6.

Создать класс **Треугольник**, заданный значениями длин трех сторон ( $a$ ,  $b$ ,  $c$ ), с методами «Периметр» и «Площадь». Определить также метод «Информация», который возвращает строку, содержащую информацию о треугольнике: длины сторон, периметр и площадь.

Создать также класс наследник **Четырехугольник**, с дополнительными параметрами – длиной четвертой стороны ( $d$ ) и длинами диагоналей ( $e$ ,  $f$ ) и переопределить методы «Периметр» (сумма всех сторон) и «Площадь».

Площадь

$$S = \frac{4e^2 f^2 - (b^2 + d^2 - a^2 - c^2)^2}{16}$$

В главной программе (либо по нажатию на кнопку) создать объект класса **Треугольник** и объект класса **Четырехугольник** и вывести информацию о них. Для упрощения проверки рекомендуется в качестве конкретного объекта класса четырехугольник взять квадрат.

### Задание 7.

Создать класс **Компьютер** со свойствами: Частота процессора (в МГц), количество ядер, объем памяти (в МБ), объем жесткого диска (в ГБ). Определить два виртуальных метода: «Стоимость», возвращающую примерную расчетную стоимость компьютера, рассчитываемую по формуле. Частота процессора \* количество ядер / 100 + количество памяти / 80 + объем жесткого диска / 20 и логический метод «Пригодность», возвращающий истину (true), если частота процессора не менее 2000 МГц, количество ядер не менее 2, объем памяти не менее 2048 МБ, и объем жесткого диска не менее 320 Гб. Определить также метод

«Информация», который возвращает строку, содержащую информацию о компьютере: частоту процессора, количество ядер, объем памяти, объем жесткого диска, стоимость и пригодность для наших нужд.

Создать также класс наследник **Ноутбук**, с дополнительным свойством.

Продолжительность автономной работы (в минутах) и переопределить методы: метод «Стоимость» возвращает число, равное стоимости обычного компьютера + количество минут автономной работы / 10, а метод «Пригодность» возвращает истину, тогда когда и ноутбук пригоден как обычный компьютер, и Продолжительность автономной работы не меньше 60 минут. В главной программе (либо по нажатию на кнопку) создать обычный **компьютер** и **ноутбук** и вывести информацию о них.

### Задание 8.

Создать класс **Прямоугольник**, заданный значениями длин двух сторон

( $a$  и  $b$ ), с виртуальными методами «Периметр» и «Площадь», возвращающими периметр и площадь соответственно, а также виртуальный метод «Увеличить в два раза», увеличивающий в два раза каждую из сторон. Определить также метод «Информация», который возвращает строку, содержащую информацию об треугольнике: длины сторон, периметр и площадь.

Создать также класс наследник **Прямоугольник со скругленными углами**, с дополнительным параметром радиус скругления ( $r$ ). Для него переопределить. Периметр по формуле  $p = 8 \cdot r + 2 \cdot \pi \cdot r$ , где  $p$  – периметр обычного прямоугольника с теми же сторонами, а Площадь по формуле  $S = 4 \cdot r^2 + \pi \cdot r^2$ , где  $S$  – площадь обычного прямоугольника. Также переопределить метод «Увеличить в два раза» так, чтобы он также увеличивал в два раза радиус скругления (по-прежнему увеличивая стороны в два раза).

В главной программе (либо по нажатию на кнопку) создать обычный **прямоугольник** и **прямоугольник со скругленными углами** и вывести информацию о них. После этого увеличить оба прямоугольника в два раза и выдать обновленную информацию.

#### Задание 9.

Создать класс **Фотоаппарат** со свойствами: Модель, Оптическое увеличение (Zoom, вещественное число от 1 до 35) и материал корпуса (металл либо пластик). Определить виртуальный метод: метод «Стоимость»

– возвращает число – стоимость фотоаппарата (в \$), рассчитываемую по формуле  $(\text{Zoom}+2) \cdot 10$ , если корпус пластиковый и  $(\text{Zoom}+2) \cdot 15$ , если материал металлический. Определить также метод «Информация», который возвращает строку, содержащую информацию об объекте: Модель, Zoom и Стоимость. Также определить логический метод «Дорогой», который будет возвращать истину (true), если стоимость фотоаппарата больше 200\$.

Создать также класс наследник **Цифровой фотоаппарат**, в котором будет дополнительный целый параметр – количество мегапикселей и переопределить метод «Стоимость», который будет возвращать число, равное стоимости обычного фотоаппарата умножить на количество мегапикселей, а также определить новый метод «Обновление модели», который увеличивает количество мегапикселей на 2.

В главной программе (либо по нажатию на кнопку) создать объект класса **Фотоаппарат** с 4-ми кратным оптическим увеличением ( $\text{Zoom}=4$ ) и пластиковым корпусом, а также **Цифровой фотоаппарат** с металлическим корпусом, 8-ю мегапикселями и 3-кратным оптическим увеличением. Вывести на экран (или форму) информацию о фотоаппаратах и о том, являются ли они дорогими. Обновить модели цифрового фотоаппарата и снова вывести информацию о нем.

### Задание 10.

Создать класс **Студент** со свойствами: ФИО, факультет, курс, минимальная оценка по экзаменам за последнюю сессию (по 5-ти бальной системе). Определить виртуальные методы: «Перевести на следующий курс», увеличивающий курс на 1, если минимальная оценка не менее 3, иначе не делающий ничего, а также «Стипендия», возвращающий стипендию (в грн): 0 грн, если минимальная оценка не выше 3, 200 грн, если минимальная оценка равна 4 и 300 грн, если минимальная оценка равна 5. Определить также метод «Информация», который возвращает строку, содержащую информацию о студенте: ФИО, факультет, курс, минимальная оценка по экзаменам и начисленную стипендию.

Создать также класс наследник **Студент-контрактник**, в котором будет дополнительный логический параметр – уплачен ли контракт и переопределены методы «Перевести на следующий курс», увеличивающий курс на 1, если минимальная оценка не менее 3 и за контракт уплачено, а также «Стипендия» возвращающий всегда 0 грн.

В главной программе (либо по нажатию на кнопку) создать объект класса **Студент** и 2 объекта класса **Студент-контрактник** (один из которых оплатил за контракт, а другой нет). Выдать информацию о студентах, затем применить к ним метод «Перевести на следующий курс» и снова выдать информацию о них.

### Задание 11.

Создать класс **Круг** заданный своим радиусом ( $r$ ), с виртуальным методом «Площадь», возвращающим площадь круга, а также виртуальный метод «Увеличить» с одним вещественным параметром – во сколько раз увеличить, увеличивающий радиус в заданное число раз. Определить также метод «Информация», который возвращает строку, содержащую информацию о круге: радиус и площадь.

Создать также класс наследник **Кольцо**, с дополнительным параметром — внутренним радиусом ( $r_{in}$ ), при этом унаследованный от родителя радиус будет обозначать внешний радиус. Переопределить метод «Площадь», как разницу между площадью внешнего круга минус площадь внутреннего круга. Также доопределить метод «Увеличить», чтобы он увеличивал также и внутренний радиус.

В главной программе (либо по нажатию на кнопку) создать обычный **круг** и **кольцо** и вывести информацию о них. После этого увеличить оба объекта в полтора раза и выдать обновленную информацию.

## Задание 12.

Создать класс **Табуретка** со свойствами: Высота ( $h$ , в см), Качество изделия (низкое, среднее, высокое). Определить два виртуальных метода: «количество древесины», которое требует табуретка, по формуле  $4 \cdot h + 12$ , если качество низкое, и  $5 \cdot h + 14$ , если качество среднее или высокое, а также «стоимость», равная  $d \cdot 2$ , для низкого качества,  $d \cdot 3$ , для среднего качества,  $d \cdot 4$ , для высокого качества, где  $d$  – количество древесины, которое требует данный объект. Определить также метод «Информация», который возвращает строку, содержащую информацию об объекте: Высоту, качество материала, количество древесины и стоимость.

Создать также класс наследник **Стул** с дополнительным свойством: высота спинки ( $h_2$ , в см), и переопределить метод «количество древесины», по формуле  $d + 2h_2 + 5$ , где  $d$  – количество древесины, которые требует табуретка с такими же параметрами (Метод «стоимость» не переопределять).

В главной программе (либо по нажатию на кнопку) создать экземпляры классов **Табуретка** и **Стул**, и напечатать информацию в таком виде:

«табуретка» + информация о табуретке и «стул» + информация о стуле

## Задание 13.

Создать класс **Фильм** со свойствами: Название, Режиссер, длительность (в минутах), количество актеров. Определить виртуальный метод:

«Стоимость», возвращающую примерную расчетную стоимость фильма (в тыс. \$), рассчитываемую по формуле  $\text{длительность} \cdot 20 + \text{количество актеров} \cdot 30$ , но если режиссер = «Стивен Спилберг» или «Джеймс Кэмерон», то стоимость в два раза выше (по сравнению с вышеуказанной формулой). Определить также метод «Информация», который возвращает строку, содержащую информацию о фильме: Название, режиссера, длительность, количество актеров и стоимость.

Создать также класс наследник **Мультфильм**, в котором переопределить метод «Стоимость» по формуле  $\text{длительность} \cdot 25 + \text{количество актеров} \cdot 10$  (вне зависимости от режиссера).

В главной программе (либо по нажатию на кнопку) создать 2 фильма с режиссерами: «Стивен Спилберг» и «Ежи Гофман», а также мультфильм и вывести информацию о них.

#### Задание 14.

Создать класс **Самолет** со свойствами: Марка, Модель, Максимальная скорость (в км/ч), Максимальная высота (в метрах). Определить виртуальный метод «Стоимость» – стоимость самолета, рассчитываемую по формуле  $\text{Максимальная скорость} * 1000 + \text{Максимальная высота} * 100$ . Определить также метод «Информация», который возвращает строку, содержащую информацию об объекте: Марка, Модель, Максимальную скорость, Максимальную высоту и Стоимость.

Создать также класс наследник **Бомбардировщик**, в котором переопределить метод «Стоимость», который вернет удвоенную стоимость относительно формулы для класса Самолет. Также создать класс **Истребитель** – наследник класса Самолет, для которого переопределить метод «Стоимость» как утроенную стоимость, относительно формулы стоимости для Самолета. В главной программе (либо по нажатию на кнопку) создать объект класса **Самолет**, класса **Бомбардировщик**, класса **Истребитель**. Вывести на экран (или форму) информацию о самолетах.

#### Задание 15.

Промоделировать отливку листов стали.

Листы стали характеризуются толщиной (в мм), и плотностью стали (в  $\text{кг/м}^3$ ).

Листы делятся на квадратные (дополнительно задаются одним числом – шириной и длиной одновременно, в мм), прямоугольные (задаются шириной и длиной, в мм) и треугольные (в виде прямоугольного треугольника, задаются двумя катетами, в мм). Для каждого типа стали определите виртуальный метод «Площадь» – возвращающий площадь листа. Также определите метод «Вес», который вычисляет вес листа, умножая площадь листа на его толщину и плотность стали. Также задайте метод «Информация», который будет выдавать информацию об листе.

В главной программе создайте массив из 15 листов стали, создав 5 квадратных, 7 прямоугольных и 3 треугольных листа случайных размеров. Выведите информацию о листах и посчитайте суммарную площадь и суммарный вес всех листов.



### Критерии оценивания.

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### Практическое занятие № 27

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Работа с типом данных структура

**Цель:** разработать структуру по индивидуальному заданию

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

#### Последовательность выполнения работы

1. Запустить MS Visual Studio
2. Создать консольное приложение
3. Выполнить задание (по вариантам)
4. Сохранить файл

#### Индивидуальные задания :

№ вар.	Поля структуры	Задача
1	Фамилия Амплуа Возраст Количество игр Количество голов	Определить лучшего форварда, и вывести сведения о футболистах, сыгравших менее 5-ти игр.
2	Фамилия Группа Физика Информ История	Определить средний бал оценок по всем предметам, и вывести сведения о студентах, средний балл которых больше 4.

3	Продавец Наименование Количество Цена Дата_продажи	Определить количество товаров, которые проданы менее года назад и вывести сведения о них.
4	Наименование Количество Цена Изготовитель Дата_поступления_на_склад	Определить количество всех товаров, количество которых больше 5 и вывести сведения об этих товарах.
5	Наименование Изготовитель Год_выпуска Количество Цена	Определить общую стоимость всех товаров, выпущенных в текущем году и вывести сведения об этих товарах.
6	Наименование Количество Цена Изготовитель Дата_выпуска	Вывести на экран наименование товара с максимальной общей стоимостью.
7	Фамилия Группа Физика Информ История	Определить средний бал оценок по физике, количество студентов с оценкой 5 по информатике и вывести сведения о них.
8	Продавец Наименование Количество Цена Дата_продажи	Определить количество товаров, проданных продавцом «Иванов», вывести сведения о них и определить товар с максимальной стоимостью.
9	Наименование Количество Цена Производитель Дата_поступления_на_склад	Вывести сведения о товарах с ценой выше средней.
10	Автор Количество страниц Тираж Год издания	Вывести данные о книгах, в которых количество страниц больше 150.

11	Автор Жанр Название Тираж	вывести данные о книгах, тираж которых не превышает 10000 экземпляров.
12	Фамилия Возраст Образование Должность	Вывести данные о работниках старших 30-ти лет, не имеющих высшего образования.
13	Фамилия Возраст Количество игр Количество пропущенных шайб	Определить средний возраст хоккеистов и вывести сведения о хоккеистах, возраст которых больше 25 лет.
14	Исполнитель Жанр Название альбома Тираж	Вывести данные о пластинках, тираж которых превышает 10000 экземпляров.
15	Производитель Объем оперативной памяти Дата изготовления Цена	Определить компьютер, изготовленный фирмой AMD с минимальной ценой и вывести все сведения о нем.
16	Фамилия Возраст Количество игр Количество заброшенных шайб	Определить средний возраст хоккеистов и вывести сведения о хоккеистах, возраст которых меньше 25 лет.
17	Наименование Производитель Год_выпуска Количество Цена	Определить общую стоимость всех товаров, выпущенных в текущем году и вывести сведения об этих товарах.
18	Наименование Количество Цена Производитель Дата выпуска	Определить среднюю стоимость товаров и товар с минимальной стоимостью.
19	Фамилия Год рождения Должность Зарплата Образование	Определить самого младшего работника и напечатать сведения о нем.

20	Фамилия Группа Год рождения оценка по физике оценка по математике оценка по информатике	Напечатать фамилии студентов, которые сдали математику на «95», и определить их количество.
21	Количество Цена Год изготовления Производитель	Определить товар, количество которого больше всего на складе, и напечатать все сведения о нем.
22	Название Частота Объем оперативной памяти Наличие DVD ROM Стоимость	Определить количество компьютеров с объемом оперативной памяти больше 10 Гбайт и напечатать все сведения о них.
23	Фамилия Группа Год рождения оценка по физике оценка по математике оценка по информатике	Определить количество студентов старше 19-ти лет, и напечатать все сведения о них.
24	Фамилия Год рождения Должность Зарплата Образование	Определить количество работников старше 60-ти лет, и напечатать все сведения о них.
25	Количество Цена Год изготовления Производитель	Определить самый дорогой товар на складе и напечатать все сведения о нем.
26	Название Частота Объем оперативной памяти Наличие DVD ROM Стоимость	Вычислить среднюю стоимость всех компьютеров и напечатать наименования компьютеров и их среднюю стоимость.
27	Фамилия Год рождения Должность Зарплата Образование	Определить количество работников - инженеров и напечатать все сведения о них.

28	Фамилия Группа Год рождения оценка по физике оценка по математике оценка по информатике	Вычислить средний балл оценок студентов по физике и напечатать фамилии, год рождения и оценки по информатике всех студентов.
29	Количество Цена Год изготовления Производитель	Определить количество товаров, произведенных более чем два года назад, и напечатать все сведения о них.
30	Название Частота Объем оперативной памяти Наличие DVD ROM Стоимость	Определить компьютеры, которые имеют DVD ROM, и напечатать все сведения о них.

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### Практическое занятие № 28

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Коллекции. Параметризованные классы.

Операции со списками.

**Цель:** овладеть навыками работы с коллекциями

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

**Последовательность выполнения работы**

1. Запустить MS Visual Studio
2. Создать консольное приложение
3. Выполнить задание (по вариантам)
4. Сохранить файл

### **Индивидуальные задания:**

Создать коллекцию согласно варианту.

№ вар.	Задача
1	«Человек»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира). Вывести сведения о самом молодом человеке.
2	«Школьник»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); школа; класс. Вывести сведения про всех учеников пятых классов.
3	«Студент»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); ВУЗ; курс; группа; средний бал; специальность. Вывести сведения про всех студентов у которых средний балл ниже 70 баллов.
4	«Покупатель»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер кредитной карточки; банковского счета. Вывести данные о покупателях с города Одессы.
5	«Пациент»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер больницы; отделение; номер медицинской карты; диагноз; группа крови. Вывести данные про пациентов с 18 отделения.

6	«Владелец автомобиля»: фамилия; имя; отчество; номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира) марка автомобиля; номер автомобиля; номер техпаспорта. Вывести данные про автомобили марки "Ваз".
7	«Военнослужащий»: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц число); должность; звание. Вывести данные про военнослужащих в звании "лейтенант".
8	«Рабочий»: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц число); № цеха; табельный номер; образование; год поступления на работу. Вывести данные про рабочих, поступивших на работу в 2010 году.
9	«Владелец телефона»: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); № телефона. Вывести данные про владельцев телефона номер, которого начинается на 720.
10	«Абитуриент»: фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); оценки по экзаменам; проходной балл. Вывести данные про абитуриентов, проходной балл которых равен больше 4 .
11	«Государство»: название страны; столица; государственный язык; население; площадь территории; денежная единица; государственный строй; глава государства. Вывести данные про государства, население которых больше 20 млн жителей.
12	«Автомобиль»: марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена. Вывести данные про автомобили, которым больше 2 лет.
13	«Товар»: наименование; стоимость; срок хранения; сорт; дата выпуска; срок годности. Вывести данные про товары срок годности которых истекает в этом году.
14	«Кинолента»: название; режиссер (фамилия; имя); год выхода; страна; стоимость; доход; прибыль. Вывести данные про фильмы режиссера Ежи Гофмана.

15	«Рейс»: марка автомобиля; номер автомобиля; пункт назначения; грузоподъемность (в тоннах); стоимость единицы груза; общая стоимость груза. Вывести данные про автомобили, грузоподъемность которых больше 2 тонн.
16	«Книга»: название; автор (фамилия; имя); год выхода; издательство; себестоимость; цена; прибыль. Вывести данные про книги авторов, фамилия которых начинается с буквы "К".
17	«Здание»: адрес; тип здания; количество этажей; количество квартир; срок эксплуатации; срок до капитального ремонта (25 лет - срок эксплуатации). Вывести данные про здания срок эксплуатации, которых больше 50 лет.
18	«Программист»: фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); образование; номер телефона. Вывести сведения о программистах, которым меньше 25 лет.
19	«Ученый»: фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); ученая степень, должность, номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира). Вывести сведения про ученых кандидатов технических наук.
20	«Пенсионер»: фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира). Вывести сведения про всех пенсионеров, которые на пенсии больше 5 лет.
21	«Футболист»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; название команды; номер в команде; амплуа; результативность (количество голов); количество игр. Вывести сведения про футболистов, которые провели за свою команду больше 50 матчей.
22	«Манекенщица»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира). Вывести данные про самую молодую манекенщицу.



23	«Международная компания»: название; интернет сайт; адрес главного офиса (почтовый индекс, страна, область, район, город, улица, дом, квартира) продолжительность пребывания на мировом рынке; количество сотрудников; количество филиалов в Европе. Вывести международные компании, количество сотрудников у которых больше 10000.
24	«Телохранитель»: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); дата рождения (год, месяц число). Вывести данные про старшего телохранителя”.
25	«Зоопарк»: Название животного; количество вида; адрес зоопарка (почтовый индекс, страна, область, район, город, улица, дом, квартира); общее количество животных, количество работников. Вывести сведения про зоопарки, в которых есть уссурийские тигры.
26	«Программное обеспечение»: название; название компании производителя; год выхода; цена. Вывести данные про программное обеспечение, которое дороже 2000 гривен.
27	«Мультфильм»: название; режиссер (фамилия; имя); год выхода; страна; стоимость; доход; прибыль . Вывести данные про мультфильмы компании “Walt Disney”.
28	«Баскетболист»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; название команды; номер в команде; амплуа; результативность (количество очков); количество игр. Вывести сведения про баскетболистов, которых забросили за свою команду больше 150 очей.
29	«Область»: название области; областной центр; население; площадь территории; губернатор. Вывести данные про области, население которых меньше 2 млн. жителей.
30	«Мотоцикл»: марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена. Вывести данные про мотоциклы марки ”Harley-Davidson”.

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 29**

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Использование регулярных выражений

**Цель:** овладеть навыками использования регулярных выражений

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

**Последовательность выполнения работы**

1. Запустить MS Visual Studio
2. Создать консольное приложение
3. Выполнить задание (по вариантам)
4. Сохранить файл

**Этапы выполнения работы:**

**Задание 1.**

В текстовом файле содержится осмысленное сообщение. Слова сообщения разделяются пробелами и знаками препинания.

1. Выведите все слова заданной длины.
2. Выведите на экран все слова сообщения, записанные с заглавной буквы.
3. Удалите из сообщения все однобуквенные слова.
4. Удалите из сообщения только те русские слова, которые начинаются на гласную букву.
5. Замените все английские слова многоточиями.
6. Найдите максимальное целое число, встречающееся в сообщении.
7. Найдите сумму всех имеющихся в тексте чисел (целых и вещественных) причем вещественное число может быть записано в экспоненциальной форме.
8. В сообщении могут встречаться номера телефонов, записанные в формате хх-хх-хх, хххххх, или ххх-хх-хх. Вывести все номера телефонов, которые содержатся в сообщении.
9. В сообщении может содержаться дата в формате дд.мм.гггг. В заданном формате дд – целое число из диапазона от 1 до 31, мм – целое число из диапазона от 1 до 12, а гггг – целое число из диапазона от 1900 до 2010 (если какая-то часть

формата нарушена, то данная подстрока в качестве даты не рассматривается). Выведите на экран все даты, которые относятся к текущему году.

10. В сообщении могут содержаться IP-адреса компьютеров в формате d.d.d.d, где d – целое число из диапазона от 0 до 255. Вывести все IP-адреса содержащиеся в тексте.

11. Выведите на экран все адреса web-сайтов, содержащиеся в сообщении.

12. В сообщении может содержаться время в формате чч:мм:сс. В заданном формате чч –целое число из диапазона от 00 до 23, мм и сс – целые числа из диапазона от 00 до 59 (если какая-то часть формата нарушена, то данная подстрока в качестве даты не рассматривается). Вывести на экран все сообщения о времени.

#### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 30**

**Тема раздела:** Объектно-ориентированное программирование

**Тема практического занятия:** Операции со списками

**Цель:** овладеть навыками использования списков

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

**Последовательность выполнения работы**

1. Запустить MS Visual Studio
2. Создать консольное приложение
3. Выполнить задание (по вариантам)

№ вар.	Задача
1	<p>Создать список с информацией о студентах:</p> <ul style="list-style-type: none"><li>– фамилия и инициалы студентов;</li><li>– номер группы;</li><li>– успеваемость (массив из трех дисциплин по 100-бальной системе);</li><li>– размер стипендии.</li></ul> <p>Вывести список студентов, имеющих по всем предметам положительные оценки и распечатать все сведения о них.</p>
2	<p>Создать список, компонентами которого является структура, содержащая следующие поля:</p> <ul style="list-style-type: none"><li>– Наименование товара;</li><li>– Стоимость единицы товара;</li><li>– Количество каждого товара.</li></ul> <p>Определить общую стоимость товара, предложенного для реализации, и его среднюю цену.</p>

3	<p>Создать список, компонентами которого является структура, содержащая следующие поля:</p> <ul style="list-style-type: none"> <li>– наименование товара;</li> <li>– стоимость товара;</li> <li>– единица измерения.</li> </ul> <p>Определить самый дорогой товар на складе и сведения о нем.</p>
4	<p>Создать список, компонентами которого является структура со следующими полями:</p> <ul style="list-style-type: none"> <li>– список работников завода;</li> <li>– должность работника;</li> <li>– заработная плата работника</li> </ul> <p>Определить среднюю зарплату слесарей - работников завода и их количество</p>

5	<p>Создать список с информацией о предложенных к реализации мониторах:</p> <ul style="list-style-type: none"> <li>– название фирмы;</li> <li>– размер по диагонали;</li> <li>– стоимость.</li> </ul> <p>Определить среднюю цену мониторов, размером не менее 19 дюймов и распечатать сведения о них.</p>
6	<p>Создать список с информацией о предложенных к реализации телевизоров:</p> <ul style="list-style-type: none"> <li>– название фирмы;</li> <li>– размер по диагонали;</li> <li>– стоимость.</li> </ul> <p>Определить количество телевизоров фирмы «Samsung», размером более 32 дюйма и распечатать сведения о них.</p>
7	<p>Создать список с информацией о наличии компьютеров для продажи:</p> <ul style="list-style-type: none"> <li>– название компьютера;</li> <li>– частота процессора;</li> <li>– объем оперативной памяти;</li> <li>– объем жесткого диска;</li> <li>– тип монитора;</li> <li>– размер монитора;</li> <li>– цена.</li> </ul> <p>Определить компьютер с наибольшей производительностью: с наибольшей скоростью процессора при объеме оперативной памяти не менее 2 Гбайт и напечатать его характеристики.</p>

8	<p>Создать список с информацией о наличии компьютеров для продажи:</p> <ul style="list-style-type: none"> <li>– название компьютера;</li> <li>– частота процессора;</li> <li>– объем оперативной памяти;</li> <li>– объем жесткого диска;</li> <li>– цена. Определить общую стоимость предложенных к продаже компьютеров, у которых частота процессора более 2 ГГц/сек фирмы «Asus» и напечатать информацию о них.</li> </ul>
9	<p>Создать список с информацией о футболистах клуба:</p> <ul style="list-style-type: none"> <li>– фамилия;</li> <li>– амплуа;</li> <li>– возраст;</li> <li>– количество игр;</li> <li>– дата проведения игры;</li> <li>– количество голов .</li> <li>– Определить лучшего форварда, который забил больше всего голов за последние 5 игр.</li> </ul>
10	<p>Создать список с информацией об авторе и его книгах:</p> <ul style="list-style-type: none"> <li>– автор;</li> <li>– название книги;</li> <li>– тираж;</li> <li>– цена экземпляра книги;</li> <li>– год издания.</li> </ul> <p>Распечатать информацию о всех авторах, которые в своих названиях используют ключевое слово «Убийство».</p>
11	<p>Создать список с информацией о работниках предприятия:</p> <ul style="list-style-type: none"> <li>– фамилия;</li> <li>– возраст;</li> <li>– образование;</li> <li>– должность;</li> <li>– пол.</li> </ul> <p>Распечатать информацию о всех работниках женского пола без высшего образования, которым в этом году необходимо оформлять пенсию.</p>

12	<p>Создать список с информацией об исполнителях джазовой музыки:</p> <ul style="list-style-type: none"> <li>– исполнитель;</li> <li>– название альбома;</li> <li>– тираж;</li> <li>– год выпуска альбома;</li> <li>– стоимость альбома.</li> </ul> <p>Распечатать сведения об исполнителе, который в текущем году выпустил альбом тиражом более 1000 экземплярах по цене не менее 50 грн.</p>
13	<p>Создать список с информацией о сотрудниках фирмы:</p> <ul style="list-style-type: none"> <li>– фамилия;</li> <li>– должность;</li> <li>– зарплата;</li> <li>– дата рождения.</li> </ul> <p>Вывести сведения о сотрудниках, у которых зарплата выше средней и возраст которых менее 30-ти лет.</p>
14	<p>Создать список с информацией о легковых автомобилях:</p> <ul style="list-style-type: none"> <li>– марка автомобиля;</li> <li>– производитель;</li> <li>– тип;</li> <li>– год выпуска;</li> <li>– стоимость.</li> </ul> <p>Вывести сведения обо всех автомобилях, срок выпуска которых не менее 5 лет и произведенных не в Китае.</p>
15	<p>Создать список с информацией о работниках телестудии:</p> <ul style="list-style-type: none"> <li>– фамилия;</li> <li>– должность;</li> <li>– образование;</li> <li>– дата приема на работу;</li> <li>– пол.</li> </ul> <p>Вывести сведения о работниках телестудии, которые работают на должности инженеров, но не имеющие высшего образования.</p>
16	<p>Создать список с информацией об автомобилях:</p> <ul style="list-style-type: none"> <li>– марка автомобиля;</li> <li>– производитель;</li> <li>– год выпуска;</li> <li>– тип двигателя;</li> <li>– общий пробег в км;</li> <li>– цена.</li> </ul> <p>Вывести сведения об автомобилях, у которых пробег составляет менее 100 км с ценой менее 200000 грн.</p>

17	<p>Создать список с информацией о работниках завода:</p> <ul style="list-style-type: none"> <li>– фамилия;</li> <li>– должность;</li> <li>– пол;</li> <li>– год рождения.</li> </ul> <p>Вывести сведения о работниках завода как для мужчин, так и для женщин, которым в текущем году предстоит оформлять пенсию.</p>
18	<p>Создать список с информацией о продовольственных товарах, хранящихся на складе:</p> <ul style="list-style-type: none"> <li>– наименование товара;</li> <li>– год поступления на склад;</li> <li>– количество;</li> <li>– производитель;</li> <li>– цена.</li> </ul> <p>Определить сумму потерь при списании товаров, срок хранения которых превышает 5 лет.</p>
19	<p>Создать список с информацией об озерах:</p> <ul style="list-style-type: none"> <li>– наименование озера;</li> <li>– страна расположения озера;</li> <li>– глубина озера;</li> <li>– соленость озера в процентах.</li> </ul> <p>Вывести сведения об озерах, глубина которых менее 50 м, а соленость более 20%.</p>
20	<p>Создать список с информацией о населенных пунктах:</p> <ul style="list-style-type: none"> <li>– название населенного пункта;</li> <li>– количество населения;</li> <li>– расстояние до почтового отделения в км.</li> </ul> <p>Вычислить средний километраж, который проходит житель населенного пункта до почтового отделения.</p>
21	<p>Создать список с информацией о реках в разных регионах страны:</p> <ul style="list-style-type: none"> <li>– наименование реки;</li> <li>– длина реки в км;</li> <li>– средняя глубина в м.</li> </ul> <p>Определить общую длину рек, у которых глубина меньше 50 м.</p>



22	<p>Создать список с информацией о клиентах кабельного телевидения:</p> <ul style="list-style-type: none"> <li>– фамилия клиента;</li> <li>– стоимость базового пакета в месяц;</li> <li>– стоимость социального пакета в месяц;</li> <li>– количество месяцев оплаты за пользование кабельным телевидением.</li> </ul> <p>Определить разницу в оплате за то количество месяцев, которое клиентом было оплачено, если бы клиент захотел перейти от базового пакета к социальному.</p>
23	<p>Создать список с информацией о футболистах клуба:</p> <ul style="list-style-type: none"> <li>– фамилия;</li> <li>– амплуа;</li> <li>– год рождения;</li> <li>– количество игр, проведенных игроком;</li> <li>– количество голов, забитых игроком.</li> </ul> <p>Вывести сведения о футболистах не старше 20 лет и забивших не менее 5 голов за сезон.</p>
24	<p>Создать список с информацией о книгах, посвященных программированию:</p> <ul style="list-style-type: none"> <li>– автор;</li> <li>– название книги;</li> <li>– количество тиража;</li> <li>– стоимость.</li> </ul> <p>Распечатать сведения о количестве книг и их общую стоимость, если книга посвящена программированию на языке C++.</p>
25	<p>Создать список с информацией о болезнях и о лекарствах:</p> <ul style="list-style-type: none"> <li>– название болезни;</li> <li>– название лекарств;</li> <li>– стоимость лекарства</li> </ul> <p>Распечатать сведения о лекарствах, способных лечить грипп или ОРЗ.</p>
26	<p>Создать список с информацией о предметах, читаемых на разных курсах:</p> <ul style="list-style-type: none"> <li>– название предмета;</li> <li>– курс, на котором читается данный предмет;</li> <li>– число часов, отводимых под данный курс.</li> </ul> <p>Распечатать информацию о курсах, читаемых на втором курсе.</p>

27	<p>Создать список с информацией о библиотеке и ее читателях:</p> <ul style="list-style-type: none"> <li>– фамилия читателя;</li> <li>– название книги, взятой читателем;</li> <li>– дата выдачи книги;</li> <li>– срок, на который выдана книга;</li> <li>– реальная дата сдачи читателем взятой книги.</li> </ul> <p>Распечатать сведения о читателях, которые не вернули книгу в указанный срок.</p>
28	<p>Создать список с информацией о молочной продукции, поступившей в магазин:</p> <ul style="list-style-type: none"> <li>– название молочного продукта;</li> <li>– дата поступления продукта в магазин;</li> <li>– срок хранения продукта по накладной;</li> <li>– дата проверки наличия продукта в магазине.</li> </ul> <p>Определить перечень продуктов, которые хранятся в магазине больше отведенного срока.</p>
29	<p>Создать список с информацией о книгах в библиотеке:</p> <ul style="list-style-type: none"> <li>– название книги;</li> <li>– жанр: для детей, для взрослых, для влюбленных, для отчаявшихся;</li> </ul> <p>Определить, какой жанр вызывает у читателей наибольший интерес.</p>
30	<p>Создать список с информацией о кинофильмах:</p> <ul style="list-style-type: none"> <li>– название фильма;</li> <li>– сколько серий;</li> <li>– длительность одной серии;</li> <li>– по каким дням проходит сериал.</li> </ul> <p>Определить, сколько дней будет демонстрироваться самый многосерийный фильм.</p>

### Практическое занятие № 31

**Тема раздела:** Паттерны проектирования

**Тема практического занятия:** Использование порождающих шаблонов, Использование порождающих шаблонов уровня архитектуры

**Цель:** изучить способы использования поведенческих шаблонов

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 180 минут

**Последовательность выполнения работы**

1. Изучить теоретическую часть, описать паттерны

2. Реализовать программный код, использующий паттерн
3. Оформит отчет

### Этапы выполнения работы:

#### Задание 1.

Написать программу битвы мага и двумера. Определить класс и тип паттерна.

#### Задание 2.

Выполнить задание согласно индивидуальному варианту. Определить класс и тип паттерна.

### Варианты индивидуальных заданий:

1	В базе данных системы электронного документооборота обрабатываются документы 3-х типов: письма, приказы и распоряжения о командировке. Каждый документ имеет номер, дату и краткую информацию о содержании. Кроме того, в письме указывается тип (входящее/исходящее) и корреспондент, в приказе –подразделение, срок выполнения и ответственный исполнитель, в распоряжении о командировке – сотрудник, период и место назначения. Вывести полный перечень документов.
2	В базе данных системы электронного документооборота обрабатываются документы 3-х типов: письма, приказы и распоряжения о командировке. Каждый документ имеет номер, дату и краткую информацию о содержании. Кроме того, в письме указывается тип (входящее/исходящее) и корреспондент, в приказе –подразделение, срок выполнения и ответственный исполнитель, в распоряжении о командировке – сотрудник, период и место назначения. Вывести содержание выбранного документа (по номеру документа).
3	В базе данных компьютерного интернет-магазина хранится номенклатура товаров 3-х типов: материнские платы, процессоры, жесткие диски. Каждый товар имеет номенклатурный номер, наименование и стоимость. Кроме того, для материнских плат указывается тип сокета, количество процессоров, тип оперативной памяти, частота системной шины, для процессоров – тип сокета, количество ядер, тактовая частота, техпроцесс, для жестких дисков – объем, скорость вращения, тип интерфейса. Вывести полную номенклатуру комплектующих.
4	В базе данных компьютерного интернет-магазина хранится

	номенклатура товаров 3-х типов: материнские платы, процессоры, жесткие диски. Каждый товар имеет номенклатурный номер, наименование и стоимость. Кроме того, для материнских плат указывается тип сокета, количество процессоров, тип оперативной памяти, частота системной шины, для процессоров – тип сокета, количество ядер, тактовая частота, техпроцесс, для жестких дисков – объем, скорость вращения, тип интерфейса. Вывести детальную информацию по товару (по номенклатурному номеру).
<b>5</b>	В информационной системе реализуется ввод и сохранение данных об абитуриентах: фамилия, имя, отчество, дата рождения, баллы за ЕГЭ (3 предмета), желательные специальности (3 специальности). Вывести всех, кто сдавал информатику.
<b>6</b>	В информационной системе реализуется ввод и сохранение данных об абитуриентах: фамилия, имя, отчество, дата рождения, баллы за ЕГЭ (3 предмета), желательные специальности (3 специальности). Вывести всех отличников.
<b>7</b>	В информационной системе реализуется ввод и сохранение данных об абитуриентах: фамилия, имя, отчество, дата рождения, баллы за ЕГЭ (3 предмета), желательные специальности (3 специальности). Вывести всех, у кого хотя бы одна 4 по одному из экзаменов.
<b>8</b>	Информационная система обеспечивает формирование отчета по лабораторным работам студента ИТ-специальности. Рассматриваются дисциплины «Базы данных», «Компьютерные сети», «Программирование». Отчет должен включать следующие разделы: титульный лист, цель работы, задание, теоретические сведения, описание экспериментальной установки, результаты работы, анализ результатов работы, выводы. Вывести номера лабораторных, где нет выводов
<b>9</b>	Информационная система обеспечивает формирование отчета по лабораторным работам студента ИТ-специальности. Рассматриваются дисциплины «Базы данных», «Компьютерные сети», «Программирование». Отчет должен включать следующие разделы: титульный лист, цель работы, задание, теоретические сведения, описание экспериментальной

	установки, результаты работы, анализ результатов работы, выводы. Вывести все данные работы
<b>10</b>	Наша «фабрика фабрик» способная производить автомобили любой марки и любого типа, готова. В будущем вы можете решить, что было бы неплохо начать выпускать внедорожники. Вам нужно будет создать ещё один интерфейс, а в офисе холдинга повесить чертёж внедорожника (добавить в CarsFactory нужный метод и реализовать его в дочерних фабриках).

#### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## **Практическое занятие № 32**

**Тема раздела:** Паттерны проектирования

**Тема практического занятия:** Использование поведенческих шаблонов.  
Использование поведенческих шаблонов уровня архитектуры

**Цель:** изучить способы использования поведенческих шаблонов

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

### **Задание 1.**

Реализовать изменения звука на магнитоле машины. Указать тип и класс паттерна

### **Задание 2.**

Реализовать использование автономных источников энергии при зарядке батареи телефона. Указать тип паттерна и класс

### **Критерии оценивания.**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 33**

**Тема раздела:** Паттерны проектирования

**Тема практического занятия:** Использование структурных шаблонов

**Цель:** изучить способы использования структурных шаблонов

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

#### **Задание 1.**

Написать программу реализующую добавление или удаление одного или двух ингредиентов в заказанное блюдо. Указать тип паттерна и класс.

#### **Задание 2.**

Реализовать картотеку архивариуса. Указать тип паттерна и класс.

#### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 34

**Тема раздела:** Оптимизация и рефакторинг

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** файл с программным кодом

**Последовательность выполнения работы**

1. Запустить MS Visual Studio
2. Создать Windows Form Application
3. Выполнить задание
4. Сохранить файл

**Задание 1.**

Создать с помощью Visual Studio консольный проект C# (согласно своему варианту), используя ориентированный подход в программировании.

**Задание 2.**

Рассчитать метрики кода средствами Visual Studio.

**Индивидуальные задания**

**Вариант 1.** Разработать приложение «Личные дела студентов». Приложение предназначено для хранения личных карточек студентов (которые содержат: пол, дату рождения, место рождения, место проживания и т. п.) в деканате и отделе кадров. Сведения должны храниться в течение всего срока обучения студентов и использоваться при составлении справок и отчетов.

**Вариант 2.** Разработать приложение «Учет успеваемости студентов». Приложение предназначено учета успеваемости студентов в сессию деканатом. Сведения об успеваемости студентов должны храниться в течение всего срока их обучения и использоваться при составлении справок о прослушанных курсах и приложений к диплому.



**Вариант 3.** Разработать приложение «Решение комбинаторно-оптимизационных задач». Приложение должно содержать алгоритмы поиска цикла минимальной длины (задача коммивояжера), поиска кратчайшего пути и поиска минимального основного дерева.

**Вариант 4.** Разработать приложение «Органайзер». Приложение предназначено для записи, хранения и поиска адресов и телефонов физических лиц и организаций, а также расписания, встреч и др. Приложение предназначено для любых пользователей компьютера.

**Вариант 5.** Разработать приложение «Калькулятор», которое предназначено для любых пользователей и должно содержать все арифметические операции (с соблюдением приоритетов) и несколько математических функций.

**Вариант 6.** Разработать приложение «Кафедра», которое содержит сведения о сотрудниках кафедры: фамилия, имя, отчество, должность, ученая степень, нагрузка, общественная работа, совместительство и др.). Приложение предназначено для использования сотрудниками отдела кадров и учебного отдела.

**Вариант 7.** Разработать приложение «Лаборатория», содержащие сведения о сотрудниках лаборатории: фамилия, имя, отчество, пол, возраст, семейное положение, наличие детей, должность, ученая степень и т. п. Приложение предназначено для использования сотрудниками профкома и отдела кадров.

**Вариант 8.** Разработать приложение «Автосервис». При записи на обслуживание заполняется заявка, в которой указываются: фамилия, имя, отчество владельца, марка автомобиля, вид работы, дата приема заказа и стоимость ремонта. После выполнения работ распечатывается квитанция.

**Вариант 9.** Разработать приложение «Учет нарушений правил дорожного движения». Для каждой автомашины (и ее владельца) в базе хранится список нарушений. Для каждого нарушения фиксируется дата, время, вид нарушения и размер штрафа. При оплате всех штрафов машина удаляется из базы.

**Вариант 10.** Разработать приложение «Агентство недвижимости», предназначенное для использования работниками агентства. В базе содержатся сведения о квартирах (количество комнат, этаж, метраж и др.). При поступлении заявки на обмен (куплю, продажу) производится поиск подходящего варианта. Если такого нет, клиент заносится в клиентскую базу и оповещается, когда вариант появляется.

**Вариант 11.** Разработать приложение «Абоненты компании сотовой связи». Картотека содержит сведения о телефонах и их владельцах. Фиксирует задолженности по оплате (абонентской и повременной). Считается, что повременная оплата местных телефонных разговоров уже введена.

**Вариант 12.** Разработать приложение «Книжный магазин», содержащее сведения о книгах (автор, название, издательство, год издания, цена). Покупатель оформляет заявку на нужные ему книги, если таковых нет, он заносится в базу и оповещается, когда нужные книги поступают в магазин.

**Вариант 13.** Разработать приложение «Авиакасса», содержащий сведения о наличии авиабилетов. В базе должны содержаться сведения о номере рейса, экипаже, типе самолета, дате и времени вылета, а также стоимости авиабилетов (разного класса). При поступлении заявки на билеты программа производит поиск подходящего рейса.

**Вариант 14.** Разработать приложение «Автостоянка», в котором содержится информация о марке автомобиля, его владельце, дате и времени въезда, стоимости стоянки, скидках, задолженности по оплате и др.

**Вариант 15.** Разработать приложение «Кадровое агентство», содержащие сведения о вакансиях и резюме. Приложение предназначено как для поиска сотрудников, отвечающих требованиям руководителей организаций, так и для поиска подходящей работы.

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

**Практическое занятие № 35**

**Тема раздела:** Оптимизация и рефакторинг кода

**Темы практических занятий:**

Оптимизация кода

Рефакторинг кода

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** файл с программным кодом

**Задание 1.**

Реализовать диаграмму классов UML, приложения реализованного в практическом занятии № 34

**Задание 2.**

Выполнить рефакторинг этого приложения используя инструменты Visual Studio.

**Задание 3.**

Рассчитать метрики кода средствами Visual Studio.

**Задание 4.**

Реализовать диаграмму классов UML приложения после выполнения рефакторинга.

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

**Практическое занятие № 36**

**Тема раздела: Разработка пользовательского интерфейса**

**Тема практического занятия:** Разработка интерфейса пользователя

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

**Последовательность выполнения работы**

1. Запустить MS Visual Studio
2. Выполнить задание
3. Сохранить файл
4. Оформить отчет о выполненной работе

**Этапы выполнения работы:****Задание**

Создать базу с 2 таблицами: родительской и дочерней. Разработать интерфейс приложения для отображения данных.

**Варианты**

Вариант 1. Предметная область Карта мира. Объекты: Страны, Города

Вариант 2. Предметная область Библиотека. Объекты: Авторы, Книги

Вариант 3. Предметная область Отдел кадров. Объекты: Подразделения, Сотрудники

Вариант 4. Предметная область Учебный отдел. Объекты: Группы, Студенты

Вариант 5. Предметная область Автосалон. Объекты: Производители автомобилей, Марки

Вариант 6. Предметная область Агентство новостей. Объекты: Категории новостей, Новости

Вариант 7. Предметная область Продуктовый магазин. Объекты: Категория продукта, Продукт

Вариант 8. Предметная область Футбол. Объекты: Команды, Игроки

Вариант 9. Предметная область Музыкальный магазин. Объекты: Исполнители, Альбомы

Вариант 10. Предметная область Аэропорт. Объекты: Авиакомпании, Рейсы

Вариант 11. Предметная область Файловая система. Объекты: Папки, Файлы

Вариант 12. Предметная область Расписание занятий. Объекты: Дни недели, Занятия

Вариант 13. Предметная область Записная книжка. Объекты: Календарные дни, Мероприятия

Вариант 14. Предметная область Видеомагазин. Объекты: Жанры, Фильмы

Вариант 15. Предметная область Железная дорога. Объекты: Дороги, Станции

Вариант 16. Предметная область Склад. Объекты: Секции, Товары

Вариант 17. Предметная область Кафедра университета. Объекты: Преподаватели, Дисциплины

Вариант 18. Предметная область Программное обеспечение. Объекты: Производители, Программные продукты

Вариант 19. Предметная область Геометрия. Объекты: Многоугольники, Вершины

Вариант 20. Предметная область Схема метро. Объекты: Линии, Станции

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## **Практическое занятие № 37-44**

**Тема раздела:** Основы ADO.Net

### **Темы практических занятий:**

Создание приложения с БД

Создание запросов к БД

Создание пользовательских форм

Создание хранимых процедур

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

### **Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

### **Этапы выполнения работы:**

#### **Задание 1.**

**Средствами ADO.Net выполнить:**

Создать базу с 2 таблицами: родительской и дочерней.  
Создать запросы к БД.  
Разработать интерфейс приложения для отображения данных.  
Создание процедуры обработки объектов формы

### **Варианты**

- Вариант 1. Предметная область Карта мира. Объекты: Страны, Города  
Вариант 2. Предметная область Библиотека. Объекты: Авторы, Книги  
Вариант 3. Предметная область Отдел кадров. Объекты: Подразделения, Сотрудники  
Вариант 4. Предметная область Учебный отдел. Объекты: Группы, Студенты  
Вариант 5. Предметная область Автосалон. Объекты: Производители автомобилей, Марки  
Вариант 6. Предметная область Агентство новостей. Объекты: Категории новостей, Новости  
Вариант 7. Предметная область Продуктовый магазин. Объекты: Категория продукта, Продукт  
Вариант 8. Предметная область Футбол. Объекты: Команды, Игроки  
Вариант 9. Предметная область Музыкальный магазин. Объекты: Исполнители, Альбомы  
Вариант 10. Предметная область Аэропорт. Объекты: Авиакомпании, Рейсы  
Вариант 11. Предметная область Файловая система. Объекты: Папки, Файлы  
Вариант 12. Предметная область Расписание занятий. Объекты: Дни недели, Занятия  
Вариант 13. Предметная область Записная книжка. Объекты: Календарные дни, Мероприятия  
Вариант 14. Предметная область Видеомагазин. Объекты: Жанры, Фильмы  
Вариант 15. Предметная область Железная дорога. Объекты: Дороги, Станции  
Вариант 16. Предметная область Склад. Объекты: Секции, Товары  
Вариант 17. Предметная область Кафедра университета. Объекты: Преподаватели, Дисциплины  
Вариант 18. Предметная область Программное обеспечение. Объекты: Производители, Программные продукты  
Вариант 19. Предметная область Геометрия. Объекты: Многоугольники, Вершины  
Вариант 20. Предметная область Схема метро. Объекты: Линии, Станции

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## МДК 01.02 Поддержка и тестирование программных модулей

### Практическое занятие № 1

**Тема раздела :** отладка программных модулей

**Цель работы:** изучить метод тестирования «Белым ящиком»

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету, MS Visual Studio

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с программным кодом

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Этапы выполнения работы:**

**Задание 1. Разработать программу на C#**

Даны длины сторон треугольника, определить вид треугольника и его площадь.

Выполнить контроль вводимых чисел.

1. Разносторонний треугольник
2. Равнобедренный треугольник
3. Равносторонний треугольник

Ограничения:

- три числа не могут быть определены как стороны треугольника;
- если хотя бы одно из них меньше или равно 0;
- сумма двух из них меньше третьего.

**Задание 2. Подготовить набор тестовых вариантов для обнаружения ошибок в программе.**

Результат оформить в следующем виде:

Таблица 1

А	В	С	Ожидаемый результат	Объект проверки
Значение	Значение	Значение	Что должно получиться	Значения вводимых данных либо ожидаемый результат

### **Задание 3. Разработать программу на C#**

Даны длины сторон треугольника, определить вид треугольника и его площадь.

Выполнить контроль вводимых чисел.

1. Остроугольный треугольник
2. Тупоугольный треугольник
3. Прямоугольный треугольник

Ограничения:

- 1) три числа не могут быть определены как стороны треугольника;
- 2) если хотя бы одно из них меньше или равно 0;
- 3) сумма двух из них меньше третьего.

Подготовить набор тестовых вариантов для обнаружения ошибок в программе и оформить результат.

**Задание 4. На основании проведенных тестов составьте рекомендации по исправлению ошибок, выявленных в ходе тестирования в виде отчета.**

Пример:

1 тест. В ходе проведения первого теста было обнаружено, что при введении некорректных данных площадь все равно высчитывается.

Рекомендуется: в случае, если пользователь введет не корректные данные, следует выводить сообщение с просьбой исправить введенные значения. Добавить в программу проверку введенных значений на соответствие ограничения.

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя



## Практическое занятие № 2

**Тема раздела :** отладка программных модулей

**Цель работы:** Разбиение на классы эквивалентности

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Этапы выполнения работы:**

**Задание**

Используя техники разбиения на эквивалентные классы и анализа граничных значений, разработать набор тестов с примерами конкретных входных данных для обеспечения достаточного тестового покрытия.

**Индивидуальные задания**

Вариант 1.

Поле "Пользователь":

- применяется маска: Фамилия\_Имя
- чувствительность к регистру: есть
- допустимые символы: [А-я], [А-з], [-\_`пробел]
- макс.длина: 256 символов

Кнопка "Сохранить". Действие по нажатию:

- при допустимом значении поля «Пользователь»: уведомление «Сохранено»
- при недопустимом значении поля «Пользователь»: уведомление «Ошибка»

Вариант 2.

Поля ввода дат "с" и "по":

- применяется маска ДД.ММ.ГГГГ
- допустимый диапазон [01.01.1900;31.12.2090]
- значение поля "с" не должно превышать значение поля "по"

Кнопка "Сохранить". Действие по нажатию:

- при допустимом значении полей ввода дат: уведомление «Сохранено»
- при недопустимом значении полей ввода дат: уведомление «Ошибка»

Вариант 3.

Поле «Температура воздуха»:

- допустимый диапазон [-50;+50]
- допустимые числа: целые и дробные с уточнением до десятых
- допустимые разделители дробной части: точка и запятая

Кнопка "Сохранить". Действие по нажатию:

- при допустимом значении поля «Температура воздуха»: уведомление «Сохранено»
- при недопустимом значении поля «Температура воздуха»: уведомление «Ошибка»

Вариант 4.

Поле «Пароль» и «Повторить пароль»:

- допустимые значения [A-z], [0-9], [\*@\$#]
- чувствительность к регистру: есть
- длина: от 8 до 32 символов

Кнопка "Сохранить". Действие по нажатию:

- уведомление «Сохранено» при одновременном выполнении условий:
  - поля заполнены допустимыми значениями
  - значения полей совпадают
- уведомление «Ошибка» при выполнении одного из условий:
  - поля заполнены недопустимыми значениями
  - значения полей не совпадают

Вариант 5.

Поле «Номер телефона»:

- допустимый формат ввода: +XXX-XX-XXX-XX-XX (без маски, т.е. запрет на ввод символов не установлен)
- пробелы обрезаются до/после строки
- допустимые символы вместо X: [0-9]

Кнопка "Сохранить". Действие по нажатию:

- при допустимом значении поля «Номер телефона»: уведомление «Сохранено»
- при недопустимом значении поля «Номер телефона»: уведомление «Ошибка»

## Вариант 6.

Поле «Электронный адрес»:

- допустимый формат ввода: ``<login>@bsu.by`` (примечание: допустимым является только домен bsu.by)
- допустимые символы для значения ``<login>``: [A-z][-\_.]
- допустимое кол-во символов для значения ``<login>``: от 2 до 50

Кнопка "Сохранить". Действие по нажатию:

- при допустимом значении поля «Электронный адрес»: уведомление «Сохранено»
- при недопустимом значении поля «Электронный адрес»: уведомление «Ошибка»

## Вариант 7.

Поле "URL":

- допустимый формат ввода: ``http[s]://<server_name>.<domain>``
- допустимые значения для переменной ``<server_name>``: [0-9], [-.], [a-z]
- допустимые значения для переменной ``<domain>``: [a-z]
- макс.длина: 256 символов

Кнопка "Сохранить". Действие по нажатию:

- при допустимом значении поля «URL»: уведомление «Сохранено»
- при недопустимом значении поля «URL»: уведомление «Ошибка»

## Вариант 8.

Поле "Возраст"

- допустимый диапазон [18-65]
- допустимые числа: целые и дробные с уточнением до 0,5
- разделитель дробной части: запятая

Кнопка "Сохранить". Действие по нажатию:

- при допустимом значении поля «Возраст»: уведомление «Сохранено»
- при недопустимом значении поля «Возраст»: уведомление «Ошибка»

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

**Практическое занятие № 3**

**Тема раздела :** отладка программных модулей

**Цель работы:** Тестовые покрытия

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Этапы выполнения работы:****Задание 1.**

Нарисовать блок -схему согласно индивидуальному заданию. Сделать тестовые покрытия, для каждого покрытия начертить граф.

**Индивидуальные задания**

№	Задача
1	Даны числа $a_1, a_2, a_3 \dots a_{10}$ , . Определить их сумму
2	Известна масса каждого из 12 предметов. Определить общую массу всего набора предметов.
3	. Известны оценки абитуриента на четырех экзаменах. Определить сумму набранных им баллов.
4	В ведомости указана зарплата, выплаченная каждому из сотрудников

	фирмы за месяц. Определить общую сумму выплаченных по ведомости денег
<b>5</b>	Известна масса каждого предмета, загружаемого в автомобиль. Определить общую массу груза.
<b>6</b>	Известно сопротивление каждого из элементов электрической цепи. Все элементы соединены последовательно. Определить общее сопротивление цепи.
<b>7</b>	Известно сопротивление каждого из элементов электрической цепи. Все элементы соединены параллельно. Определить общее сопротивление цепи.
<b>8</b>	Известны оценки по физике каждого из 20 учеников класса. Определить среднюю оценку.
<b>9</b>	Известны оценки ученика по 10 предметам. Определить среднюю оценку
<b>10</b>	Известны оценки по алгебре каждого ученика класса. Определить среднюю оценку
<b>11</b>	Известна масса каждого предмета из некоторого набора предметов. Определить среднюю массу
<b>12</b>	Известны оценки двух учеников по четырем предметам. Определить сумму оценок каждого ученика.
<b>13</b>	Известны результаты двух спортсменов-пятиборцев в каждом из пяти видов спорта в баллах. Определить сумму баллов, полученных каждым спортсменом.
<b>14</b>	Известен возраст (в годах в виде 14,5 лет и т. п.) каждого ученика двух классов. Определить средний возраст учеников каждого класса. В каждом классе учатся 20 человек.
<b>15</b>	Известно количество осадков, выпавших за каждый день января и

	марта. Определить среднее количество осадков за каждый месяц
<b>16</b>	Известен рост каждого ученика двух классов. Определить средний рост учеников каждого класса. Численность обоих классов одинаковая.
<b>17</b>	Известны оценки по физике каждого ученика двух классов. Определить среднюю оценку в каждом классе. Количество учащихся в каждом классе одинаковое.
<b>18</b>	Напечатать таблицу стоимости 50, 100, 150, ..., 1000 г сыра (стоимость 1 кг сыра вводится с клавиатуры).
<b>19</b>	Напечатать таблицу стоимости 100, 200, 300, ..., 2000 г конфет (стоимость 1 кг конфет вводится с клавиатуры).
<b>20</b>	Одна штука некоторого товара стоит 20,4 руб. Напечатать таблицу стоимости 2, 3, ..., 20 штук этого товара.
<b>21</b>	Напечатать таблицу соответствия между весом в фунтах и весом в килограммах для значений 1, 2, ..., 10 фунтов (1 фунт = 453 г).
<b>22</b>	Напечатать таблицу перевода расстояний в дюймах в сантиметры для значений 10, 11, ..., 22 дюйма (1 дюйм = 25,4 мм).
<b>23</b>	Напечатать таблицу перевода 1, 2, ... 20 долларов США в рубли по текущему курсу (значение курса вводится с клавиатуры).
<b>24</b>	Считая, что Земля — идеальная сфера с радиусом $R = 6350$ км, определить расстояние до линии горизонта от точки с высотой над Землей, равной 1, 2, ... 10 км.
<b>25</b>	Одна штука некоторого товара стоит 20,4 руб. Напечатать таблицу стоимости 2, 3, ..., 20 штук этого товара.
<b>26</b>	Даны вещественные числа $a, b, c$ ( $a \neq 0$ ). Выяснить, имеет ли квадратное уравнение с данными параметрами решение
<b>27</b>	Известны площади круга и квадрата. Определить: уместится ли круг в квадрате?

<b>28</b>	Известны площади круга и квадрата. Определить: уместится ли квадрат в круге?
<b>29</b>	Известны площади круга и равностороннего треугольника. Определить: уместится ли круг в треугольнике?
<b>30</b>	Известны площади круга и равностороннего треугольника. Определить: уместится ли треугольник в круге?

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 4**

**Тема раздела :** виды тестирования

**Цель работы:** разработка тестовых сценариев

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Этапы выполнения работы:**

**Задание № 1.** Написать программу решения квадратного уравнения

$$ax^2 + bx + c = 0.$$

**Задание № 2.** Найти минимальный набор тестов для программы нахождения вещественных корней квадратного уравнения  $ax^2 + bx + c = 0$ .

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

**Практическое занятие № 5**

**Тема раздела :** виды тестирования

**Цель работы:** тестирование черным ящиком

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Этапы выполнения работы:**

**Задание.** Реализовать программу на формах, согласно индивидуальному варианту.

Выполнить следующий набор тестов :

- 1) заполнение букв -не выполнено
- 2) заполнение отрицательных элементов-выполнено
- 3) – ручной ввод значений – выполнено
- 4) Заполнение автоматическое - выполнено

**Индивидуальные задания:**

Вариант 1. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить сумму элементов массива, расположенных после минимального элемента.



Вариант 2. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить сумму элементов массива, расположенных между минимальным и максимальным элементами.

Вариант 3. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить среднеарифметическое значение элементов массива.

Вариант 4. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить среднеквадратическое значение элементов массива.

Вариант 5. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить среднегеометрическое значение ненулевых элементов массива.

Вариант 6. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить среднегармоническое значение положительных элементов массива.

Вариант 7. В одномерном динамическом массиве, состоящем из  $p$  элементов, поменять местами максимальный и минимальный элементы.

Вариант 8. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить сумму элементов массива, расположенных между первым и последним отрицательными элементами.

Вариант 9. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить сумму элементов массива, расположенных до последнего положительного элемента.

Вариант 10. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить сумму элементов массива, расположенных между первым и последним нулевыми элементами.

Вариант 11. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить сумму модулей элементов массива, расположенных после первого элемента, равного нулю.

Вариант 12. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить сумму положительных элементов массива, расположенных до максимального элемента.

Вариант 13. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить сумму элементов массива, расположенных до минимального элемента.

Вариант 14. В одномерном динамическом массиве, состоящем из  $p$  элементов, вычислить сумму элементов массива, расположенных между первым и последним положительными элементами.

Вариант 15. В одномерном динамическом массиве, состоящем из  $n$  элементов, вычислить среднее значение элементов, расположенных в массиве между первым последним нулевыми элементами.

Вариант 16. В одномерном динамическом массиве, состоящем из  $n$  элементов, вычислить произведение элементов массива, расположенных между первым и вторым нулевыми элементами.

Вариант 17. В одномерном динамическом массиве, состоящем из  $n$  элементов, определить номер минимального и максимального элементов массива.

Вариант 18. В одномерном динамическом массиве, состоящем из  $n$  элементов, определить сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Вариант 19. В одномерном динамическом массиве, состоящем из  $n$  элементов, определить количество элементов массива, больших  $C$ .

Вариант 20. В одномерном динамическом массиве, состоящем из  $n$  элементов, определить количество элементов массива, меньших  $C$ .

Вариант 21. В одномерном динамическом массиве, состоящем из  $n$  элементов, вычислить сумму отрицательных элементов массива.

Вариант 22. В одномерном динамическом массиве, состоящем из  $n$  элементов, вычислить сумму положительных элементов массива.

Вариант 23. В одномерном динамическом массиве, состоящем из  $n$  элементов, вычислить произведение элементов массива с четными номерами.

Вариант 24. В одномерном динамическом массиве, состоящем из  $n$  элементов, вычислить сумму элементов массива, расположенных после минимального элемента.

Вариант 25. В одномерном динамическом массиве, состоящем из  $n$  элементов, вычислить сумму элементов массива, расположенных до максимального элемента.

Вариант 26. В одномерном динамическом массиве, состоящем из  $n$  элементов, выполнить поиск элемента по заданному ключу последовательным методом поиска.

Вариант 27. В одномерном динамическом массиве, состоящем из  $n$  элементов, выполнить поиск элемента по заданному ключу бинарным методом поиска.

Вариант 28. В одномерном динамическом массиве, состоящем из  $p$  элементов, выполнить поиск элемента по заданному ключу интерполяционным методом поиска.

Вариант 29. В одномерном динамическом массиве, состоящем из  $p$  элементов, определить количество и индексы элементов массива, больших  $C$ .

Вариант 30. В одномерном динамическом массиве, состоящем из  $p$  элементов, определить количество и индексы элементов массива, меньших  $C$ .

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 6

**Тема раздела :** виды тестирования

**Цель работы:** разработка тестовых пакетов

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Этапы выполнения работы:**

**Задание.** Реализовать программу согласно индивидуальному заданию.

Закодировать сообщение, используя код Полибия

Древнегреческий историк и полководец Полибий придумал сложный способ шифрования, названный в честь его квадрат Полибия. «Квадрат Полибия» представляет собой квадрат 6х6 (5х5), столбцы и строки которого нумеруются цифрами от 1 до 6 (5). Все буквы алфавита вписывались в квадрат по одной на клетку. (можно в одну клетку поместить буквы е-э, и-й, ж-з, р-с, ф-х, ш-щ). Буквы расположены в алфавитном порядке.

В результате каждой букве соответствует пара чисел, и зашифрованное сообщение превращается в последовательность пар чисел. Расшифровывается путем нахождения буквы стоящей на пересечении строки и столбца.

Суть в том, что каждой букве предназначена своя пара цифр одна по горизонтальной линии, а другая по вертикальной. Данный вид кодирования изначально применялся для греческого алфавита, но затем был распространен на другие языки.

	1	2	3	4	5	6
1	<b>А</b>	<b>Б</b>	<b>В</b>	<b>Г</b>	<b>Д</b>	<b>Е</b>
2	<b>Ё</b>	<b>Ж</b>	<b>З</b>	<b>И</b>	<b>Й</b>	<b>К</b>
3	<b>Л</b>	<b>М</b>	<b>Н</b>	<b>О</b>	<b>П</b>	<b>Р</b>
4	<b>С</b>	<b>Т</b>	<b>У</b>	<b>Ф</b>	<b>Х</b>	<b>Ц</b>
5	<b>Ч</b>	<b>Ш</b>	<b>Щ</b>	<b>Ъ</b>	<b>Ы</b>	<b>Ь</b>
6	<b>Э</b>	<b>Ю</b>	<b>Я</b>	<b>,</b>	<b>.</b>	<b>-</b>

### Индивидуальные задания

№ варианта	Задание
1	Если бы да кабы, да во рту росли грибы.
2	Конь о четырех ногах, а спотыкается.
3	Дают — бери, а бьют — беги.
4	Кто рано встает, тому Бог подает.
5	Взялся за гуж — не говори, что не дюж.
6	На Бога надейся, а сам не плошай.
7	Поспешешь — людей насмешишь.
8	Слово — не воробей, вылетит — не поймаешь.
9	Обещать — дело господское, исполнять — холопское.
10	Дурная голова ногам покоя не дает.
11	Мягко стелет, да жестко спать.
12	Бей своих, чтобы чужие боялись.
13	Не плюй в колодезь — пригодится напиток.
14	Куй железо, пока горячо.
15	Хлеба ни куска, и стол — доска
16	Хлеб наш насущный: хоть черный, да вкусный
17	Хлеб ногами топтать — народу голодать
18	Без ума проколотишься, а без хлеба не проживешь
19	Блюда хлеб на обед, а слово — на ответ
20	Весною сутки мочит, а час сушит.

<b>21</b>	Вешний ледок, что чужой избы порог.
<b>22</b>	Вешняя пора - поел да и со двора.
<b>23</b>	Кто спит весною - плачет зимою.
<b>24</b>	Прилетела бы чайка, а весна будет.
<b>25</b>	Матушка – весна всем красна.
<b>26</b>	Солнце светит, солнце сияет – вся природа воскресает.
<b>27</b>	Готовь сани с весны, а колёса с осени.
<b>28</b>	Летний день на вес золота, он дороже зимней недели.
<b>29</b>	Лето - отрадная пора, с курами ложись, а с петухами на работу вставай.
<b>30</b>	Летом дома сидеть — зимой хлеба не иметь.

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## **Практическое занятие № 7**

**Тема раздела :** виды тестирования

**Цель работы:** Использование инструмента анализа качества

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Этапы выполнения работы:**

**Задание 1.** Написать программу, генерирующую массив вещественных чисел в диапазоне от  $-10$  до  $10$  и определяющую все минимальные положительные элементы.

**Задание 2.** Оценить качество разработанной программы

	Правильность	Универсальность	Проверяемость	Точность результатов
Недостатки				
Оценка				

**Задание 3.** Улучшить программу

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 8

**Тема раздела :** Тестовый дизайн

**Цель работы:** Создание тестовых кейсов

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Этапы выполнения работы:**

**Задание 1.** Ознакомьтесь с шаблон тестового кейса

**Аннотация теста**

<b>Название проекта</b>	
<b>Рабочая версия</b>	
<b>Имя тестирующего</b>	
<b>Дата(ы) теста</b>	

**Расшифровка тестовых информационных полей:**

Поле	Описание
<b>Название проекта</b>	Название тестируемого проекта
<b>Рабочая версия</b>	Версия проекта/программного обеспечения (первый тест считается 1.0).
<b>Имя тестирующего</b>	Имя того, кто проводил тесты
<b>Дата(ы) теста</b>	Дата(ы) проведения тестов – это один или несколько дней. Если тесты проводились в более протяженный период времени, нужно отметить отдельную дату для каждого теста.
<b>Тестовый пример #</b>	Уникальный ID для каждого



	тестового примера. Следуйте некоторым конвенциям, чтобы указать типы тестов. Например, 'ТС_UI_1' означает 'user interface test case #1' (ТС_ПИ_1: тестовый случай пользовательского интерфейса#1)
<b>Приоритет тестирования</b> (Низкий/Средний/Высокий)	Насколько важен каждый тест. Приоритет тестирования для бизнес-правил и функциональных тестовых случаев может быть средним или высоким, в то время как незначительные случаи пользовательского интерфейса могут иметь низкий приоритет.
<b>Заголовок/название теста</b>	Название тестового случая. Например, Подтвердите страницу авторизации с действительным именем пользователя и паролем.
<b>Краткое изложение теста</b>	Описание того, что должен достичь тест.
<b>Этапы теста</b>	Перечислите все этапы теста подробно. Запишите этапы теста в том порядке, в котором они должны быть реализованы. Предоставьте как можно больше подробностей и разъяснений. Пронумерованный список – хорошая идея.
<b>Тестовые данные</b>	Перечислите/опишите все тестовые данные, используемые для данного тестового случая. Так, фактические используемые входные данные можно отслеживать по результатам тестирования. Например, Имя пользователя и пароль для подтверждения входа.
<b>Ожидаемый результат</b>	Каким должен быть вывод системы после выполнения

	теста? Подробно опишите ожидаемый результат, включая все сообщения/ошибки, которые должны отображаться на экране.
<b>Фактический результат</b>	Каким должен быть фактический результат после выполнения теста? Опишите любое релевантное поведение системы после выполнения теста.
<b>Предварительное условие</b>	Любые предварительные условия, которые должны быть выполнены до выполнения теста. Перечислите все предварительные условия для выполнения этого тестового случая.
<b>Постусловие</b>	Каким должно быть состояние системы после выполнения теста?
<b>Статус</b> <i>(Зачет/Незачет)</i>	Если фактический результат не соответствует ожидаемому результату, отметьте тест как неудачный. В ином случае обновление пройдено.
<b>Примечания/комментарии</b>	Используйте эту область для любых дополнительных замечаний/комментариев/вопросов. Эта область предназначена для поддержки вышеуказанных полей (например, если есть некоторые особые условия, которые не могут быть описаны в любом из вышеуказанных полей, или если есть вопросы, связанные с ожидаемыми или фактическими результатами).

**Задание 2.** Используя шаблон и задачу из практического задания № 5 составьте тестовый кейс, согласно индивидуальному заданию

Тестовый пример #1:

Тестовый пример #	
Приоритет тестирования	
Заголовок/название теста	
Краткое изложение теста	
Этапы теста	
Тестовые данные	
Ожидаемый результат	
Фактический результат	
Статус	
Предварительное условие	
Постусловие	
Примечания/комментарии	

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## **Практическое занятие № 9**

**Тема раздела :** Тестовый дизайн

**Цель работы:** Создание тестового плана

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Этапы выполнения работы:**

**Задание.** Используя задачу из практического задания № 5 составить тестовый план для каждого компонента программы.

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 10

**Тема раздела :** Тестовый дизайн

**Цель работы:** Применение техник тестового дизайна

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Задание.** Используя следующие техники:

- 1) Эквивалентное Разделение (Equivalence Partitioning), далее в тексте - EP
- 2) Анализ Граничных Значений (Boundary Value Analysis), далее в тексте - BVA
- 3) Предугадывание ошибки (Error Guessing), далее в тексте - EG
- 4) Причина / Следствие (Cause/Effect), далее в тексте - CE

План разработки тест кейсов предлагается следующий:

1. Анализ требований.
2. Определение набора тестовых данных на основании EP, BVA, EG.
3. Разработка шаблона теста на основании CE.
4. Написание тест кейсов на основании первоначальных требований, тестовых данных и шагов теста.

Протестировать функциональность формы приема заявок, требования к которой предоставлены в следующей таблице:

Элемент	Тип элемента	Требования
Тип обращения	combobox	Набор данных: 1. Консультация 2. Проведение тестирования 3. Размещение рекламы 4. Ошибка на сайте * - на процесс выполнения операции приема заявок не влияет.
Контактное лицо	editbox	1. Обязательное для заполнения 2. Максимально 25 символов

		3. Использование цифр и спец символов не допускается
Контактный телефон	editbox	<ol style="list-style-type: none"> <li>1. Обязательное для заполнения</li> <li>2. Допустимые символы "+" и цифры</li> <li>3. "+" можно использовать только в начале номера</li> <li>4. Допустимые форматы: <ul style="list-style-type: none"> <li>• начинается с плюса - 11-15 цифр</li> <li>• +31612361264</li> <li>• +375291438884</li> <li>• без плюса - 5-10 цифр, например:</li> <li>• 0613261264</li> <li>• 2925167</li> </ul> </li> </ol>
Сообщение	text area	<ol style="list-style-type: none"> <li>1. Обязательное для заполнения</li> <li>2. Максимальная длина 1024 символа</li> </ol>
Отправить	button	<p>Состояние:</p> <ol style="list-style-type: none"> <li>1. По умолчанию - не активна (Disabled)</li> <li>2. После заполнения обязательных полей становится активна (Enabled)</li> </ol> <p>Действия после нажатия</p> <ol style="list-style-type: none"> <li>1. Если введенные данные корректны - отправка сообщения</li> <li>2. Если введенные данные НЕ корректны - валидационное сообщение</li> </ol>

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 11

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Модульное тестирование

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Задание.** Разработать программу для подсчета объема цилиндра и создать модульный тест.

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 12

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Unit тесты

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Задание.** Разработать вычитание и умножение для калькулятора. Протестировать модульными тестами.

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

**Практическое занятие № 13**

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Обработка исключительных ситуаций

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 90 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Задание.** Разработать программу, в которой обрабатываются следующие исключительные ситуации:

- 1) Отрицательное значение возраста
- 2) Год рождения больше текущей даты

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя



## Практическое занятие № 14

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Функциональное тестирование

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

4. Выполнить задание
5. Сохранить файл
6. Оформить отчет о выполненной работе

**Задание 1.** Пусть необходимо выполнить тестирование программы, определяющей точку пересечения двух прямых на плоскости. Попутно, она должна определять параллельность прямой одной из осей координат.

**Задание 2.** Опишите методы формирования тестового кейса:

- 1) Анализ граничных значений
- 2) Эквивалентные классы разбиения

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 15

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Нагрузочное тестирование

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

7. Выполнить задание
8. Сохранить файл
9. Оформить отчет о выполненной работе

**Задание 1.** Разработать Компилятор простых арифметических выражений, например  $2 + 4 + (-5) * (7 - 8)$ . Вход и выход осуществляются в виде строк

**Задание 2.** Разработать тестовый сценарий нагрузочного тестирования.

Ответить на вопрос – сколько запросов в секунду может обработать приложение при условии, что они идут последовательно. Построить график зависимости времени ответа от количества параллельных запросов (рассматривать логарифмическую шкалу по основанию два, т.е. 1, 2, 4, 8, 16, 32 и т.д. запроса)

Ответить на вопрос – какое максимальное количество параллельных запросов может обработать приложение без сбоев.

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 16

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Интеграционное тестирование

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 240 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Задание № 1.** Разработать приложение, состоящее из трех модулей:

- ♣ главный модуль, считывающий из текстового файла координаты точек на плоскости;
- ♣ модуль, содержащий функции расчета расстояния между двумя точками;
- ♣ модуль, содержащий функцию, определяющую треугольник с максимальной площадью.

**Задание № 2.** Описать этапы нисходящего проектирования разработанного приложения.

**Задание № 3.** Описать этапы восходящего проектирования разработанного приложения

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## **Практическое занятие № 17**

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Конфигурационное тестирование

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Задание № 1.** Протестировать хранитель экрана на различных конфигурациях.  
Составить отчет о дефектах

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 18

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Веб-тестирование

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 240 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Задание № 1.** Протестировать сайт, согласно плану

### План тестирования

1. Что надо тестировать: Сайт — <https://www.ozon.ru/>
2. Что будем тестировать:
  - Регистрация;
  - Авторизацию;
  - Поиск;
  - Добавление товара в корзину.
3. Как будем тестировать: Функциональное тестирование
4. Окружение: ОС Windows 10 x64, браузер любой
5. Тестирование проводится на готовом продукте
6. Риски: Отсутствуют

**Задание 2.** Составить тестовые кейсы, согласно шаблону

Тест-кейс №1

Поле	Описание	
Идентификатор (ID)	Test_keis_1	
Приоритет (Priority)		
Связанное требование (Related task)		
Название (Summary)		
Предусловия (Preconditions)		
Шаги воспроизведения (Steps to reproduce)		
Ожидаемый результат (Expected Result)	Значение	Результат

Фактический результат	Значение	Результат

### **Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **Практическое занятие № 19**

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** создание тестовых документов

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

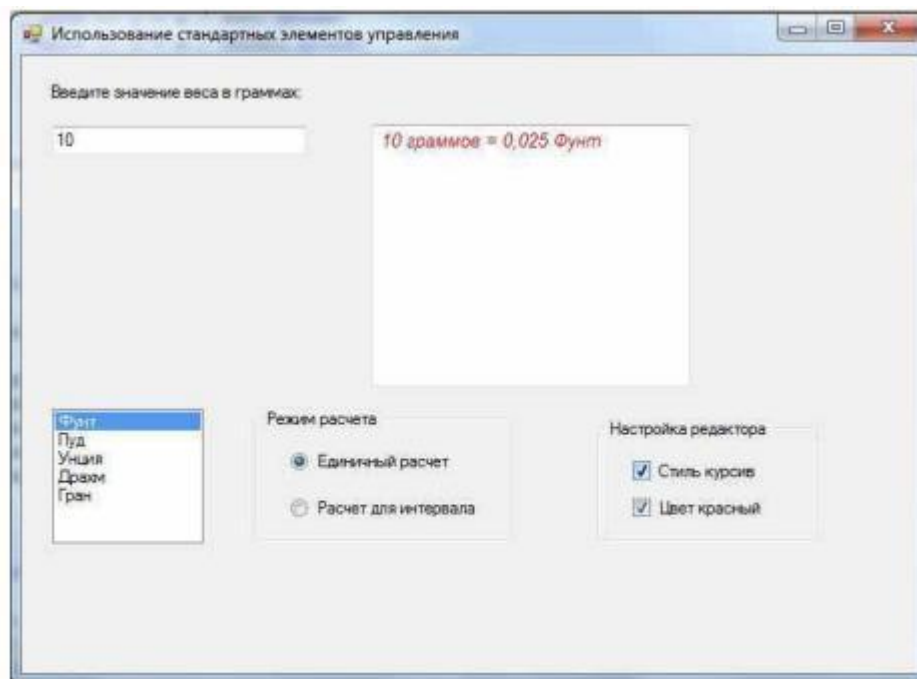
**Время выполнения:** 240 минут

**Форма отчетности по занятию:** файл с выполненной работой

#### **Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

## Задание 1. Создать приложение, представленное на рисунке



**Задание 2.** Создать тетовый план, отчет о дефектах и диаграмму готовности программного модуля.

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 20

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Системное тестирование

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 240 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выполнить задание
2. Сохранить файл
3. Оформить отчет о выполненной работе

**Задание № 1.** Дана структура с именем ZNAK, состоящая из полей: – фамилия, имя; – знак Зодиака; – дата рождения (массив из трех чисел). Написать программу, которая выполняет следующие действия:

– ввод с клавиатуры данных в массив, состоящий из 8 элементов типа ZNAK, и занесение их в файл данных;

– чтение данных из файла и вывод их на экран; – вывод на экран информации о людях, родившихся в месяц, значение которого введено с клавиатуры (если таких нет – вывести об этом сообщение);

– список должен быть упорядочен по знакам Зодиака.

**Задание № 2.** Описать и обосновать итоги тестирования работы разработанного приложения на различных платформах: различных вариантах аппаратной конфигурации, версиях операционной системы и окружения

**Критерии оценивания:**

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 несущественных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя



## Практическое занятие № 21

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Usability тестирование

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,


**Время выполнения:** 240 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Последовательность выполнения работы**

1. Выбрать программный продукт для исследования его целевой аудитории. Составить общую характеристику целевой аудитории.
2. Сегментировать целевую аудиторию в зависимости от параметров, наиболее влияющих на проектируемую систему (пол, возраст, род занятий, уровень компьютерной грамотности и т.п.).
3. В каждом сегменте выделить типичную персону.
4. Разработать карту эмпатии для этой персоны
5. Разработать профиль «ключевого персонажа» и «второстепенного персонажа» согласно таблице 1:

Таблица 1 – Профиль персонажа

<Категория целевой аудитории>	
<Роль персоны>	
	<b>Описание</b> <В описании указываются следующие характеристики: ФИО, пол, возраст, род занятий, семейное положение, образование, увлечения, социальный статус, место работы>
Фотография	
<b>Личные характеристики</b>	
<b>Цели:</b> <Перечень целей, которые пользователь стремится достичь во время использования системы>	
<b>Взаимодействие с продуктом</b> <Рабочий процесс и контекст (окружение)>	
<b>Неудовлетворенности и ожидания</b> <описание исключительных ситуаций и вытекающих из них проблем, описание дополнительных возможностей программного продукта>	

6. Разработать сценарий взаимодействия персонажа с программным продуктом.
7. Составить карту сценариев на основании таблицы 2.

Таблица 2 – Карта сценариев

Шаг:	<название шага 1>
Вопросы:	<перечислить все вопросы, которые могут возникнуть у персонажа при работе с продуктом на данном шаге>
Пожелания:	<перечислить пожелания, которые могут возникнуть у персонажа при работе с продуктом на данном шаге >
Эмоции:	<какие эмоции возникнут у персонажа от взаимодействия с продуктом>
Комментарии:	<описание окружения, исходных данных>

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 незначительных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

## Практическое занятие № 22

**Тема раздела :** Отдельные виды тестирования

**Цель работы:** Usability тестирование

**Материально-техническое и комплексно-методическое обеспечение:** Для проведения практической работы используется следующее обеспечение: персональный компьютер, подключённый к Интернету,

**Время выполнения:** 180 минут

**Форма отчетности по занятию:** файл с выполненной работой

**Задание № 1.** Изучите и опишите одно из средств выявления уязвимостей

### Критерии оценивания:

**Оценка 5 «отлично»** работа выполнена полностью и правильно, сделаны правильные выводы

**Оценка 4 «хорошо»** работа выполнена правильно с учетом 1-2 незначительных ошибок, исправленных самостоятельно по требованию преподавателя

**Оценка 3 «удовлетворительно»** работа выполнена правильно не менее чем на половину или допущены 3-4 существенные ошибки

**Оценка 2 «неудовлетворительно»** допущены 5 и более существенные ошибки в ходе работы, которые обучающийся не может исправить даже по требованию преподавателя

### **МДК. 01.03 Разработка мобильных приложений Практическое занятие № 1**

**Тема раздела:** Классификация мобильных устройств.

**Тема практического занятия:** Знакомство с платформой. Принципы работы.

**Цель:** ознакомиться с платформой.

**Планируемые результаты:**

**уметь:** работать с платформой.

**знать:** платформу и принципы работы.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Написать консольное приложение производящий арифметические операции с двумя числами. Приложение должно быть разработано с использованием форм Visual Studio.

**Задание 2.** Написать консольное приложение позволяющее работать с графиками на координатной плоскости.

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 2

**Тема раздела:** Классификация мобильных устройств.

**Тема практического занятия:** Создание эмуляторов и подключение устройств.

**Цель:** создание эмуляторов и подключение устройств.

**Планируемые результаты:**

**уметь:** создавать эмуляторы и подключение устройств.

**знать:** создание эмуляторов и подключение устройств.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Научиться писать обработчики событий для объектов класса System.Windows.Forms.Form.

**Задание 2.** Исследовать события KeyDown, KeyUp, KeyPress, происходящие при вводе данных с клавиатуры.

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

### **Практическое занятие № 3**

**Тема раздела:** Классификация мобильных устройств.

**Тема практического занятия:** Создание виртуального устройства первого приложения.

**Цель:** создать визуальное устройство первого приложения.

**Планируемые результаты:**

**уметь:** создавать визуальное устройство первого приложения.

**знать:** создание визуального устройства первого приложения.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Разработать мобильное приложение игра «Три в ряд»

**Задание 2.** Разработать мобильное приложение «Календарь»

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 4

**Тема раздела:** Классификация мобильных устройств.

**Тема практического занятия:** Разработка мобильных приложений на примере XamarinStudio.

**Цель:** ознакомиться с работой БД.

**Планируемые результаты:**

**уметь:** работать с БД.

**знать:** БД.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** В XamarinStudio разработать мобильное приложение для регистрации пользователей. Регистрация должна включать в себя следующие типы данных: пароль, логин и номер телефона.

**Задание 2.** В XamarinStudio разработать мобильное приложение «Заметки».

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## **Практическое занятие № 5**

**Тема раздела:** Проектирование и отладка мобильных приложений

**Тема практического занятия:** Изучение и комментирование кода.

Изменение элементов дизайна

**Цель:** изучить элементы дизайна и комментирование кода

**Планируемые результаты:**

**уметь:** создавать элементы дизайна и комментировать код.

**знать:** элементы дизайна в создании мобильных приложений.

Комментирование кода.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:**

- методические рекомендации к выполнению работы;
- задание ;
- компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Разместите на форме две кнопки (Кнопка) и одну метку (Метка). Сделайте на кнопках следующие надписи: «привет», «до свидания». Создайте обработчики события нажатия на данные кнопки, которые будут менять текст метки на слова, написанные на кнопках. Создайте обработчик события создания формы (Нагрузка), который будет устанавливать цвет формы и менять текст метки на строку «Начало работы».

**Задание 2.** Разместите на форме ряд кнопок (Кнопка) напротив каждой поле ввода (Текстовое поле) и одну метку (Метка). Создайте обработчики события нажатия на данные кнопки, которые будут менять текст в метке. Текст в метке берется из поля ввода напротив нажимаемой кнопки.

**Задание 3.** Разместите на форме ряд кнопок (Кнопка), и одно поле ввода (Текстовое поле). Создайте обработчики события нажатия на данные кнопки, которые будут менять текст на нажатой кнопке. Текст на кнопке берется из поля ввода.

**Задание 4.** Разместите на форме ряд кнопок (Кнопка) и ряд меток (Метка). Создайте обработчики события нажатия на данные кнопки, которые будут менять цвет двух меток. Создайте обработчик события нажатия кнопки мыши на форме (Щелчок), который будет устанавливать цвет всех меток в белый.

**Задание 5.** Разместите на форме две кнопки (Кнопка) и одно поле ввода (Текстовое поле). Сделайте на кнопках следующие надписи: «заполнить», «очистить». Создайте обработчики события нажатия на данные кнопки, которые будут очищать или заполнять поле ввода знаками «\*\*\*\*\*».



Создайте обработчик события создания формы (Нагрузка), который будет устанавливать цвет формы и менять текст в поле ввода на строку «+++++».

**Задание 6.** Разработайте игру, которая заключается в следующем. На форме размещены пять кнопок (Кнопка). При нажатии на кнопку некоторые кнопки становятся видимыми, а другие - невидимыми. Цель игры – скрыть все кнопки.

**Задание 7.** Разработайте игру, которая заключается в следующем. На форме размещены четыре кнопки (Кнопка) и четыре метки (Метка). При нажатии на кнопку часть надписей становится невидимой, а часть, наоборот, становится видимой. Цель игры - скрыть все надписи.

**Задание 8.** Разместите на форме ряд кнопок (Кнопка). Создайте обработчики события нажатия на данные кнопки, которые будут делать неактивными текущую кнопку. Создайте обработчик события изменения размера формы (Изменение размера), который будет устанавливать все кнопки в активный режим.

**Задание 9.** Разместите на форме ряд кнопок (Кнопка). Создайте обработчики события нажатия на данные кнопки, которые будут делать неактивными следующую кнопку. Создайте обработчик события нажатия кнопки мыши на форме (Щелчок), который будет устанавливать все кнопки в активный режим.

**Задание 10.** Разместите на форме три кнопки (Кнопка) и одно поле ввода (Текстовое поле). Сделайте на кнопках следующие надписи: «\*\*\*\*\*», «+++++», «00000». Создайте обработчики события нажатия на данные кнопки, которые будут выводить текст, написанный на кнопках, в поле ввода. Создайте обработчик события создания формы (Нагрузка), который будет устанавливать цвет формы и менять текст в поле ввода на строку «Готов к работе».

**Задание 11.** Разместите на форме ряд полей ввода (текстовое поле). Создайте обработчики события нажатия кнопкой мыши на данные поля ввода, которые будут выводить в текущее поле ввода его номер. Создайте обработчик события изменения размера формы (Изменение размера), который будет очищать все поля ввода.

**Задание 12.** Разместите на форме поле ввода (текстовое поле), метку (Метка) и кнопку (Кнопка). Создайте обработчик события нажатия на кнопку, который будет копировать текст из поля ввода в метку. Создайте обработчик события нажатия кнопки мыши на форме (Щелчок), который будет устанавливать цвет формы и менять текст метки на строку «Начало работы» и очищать поле ввода.

**Задание 13.** Разместите на форме поле ввода (текстовое поле) и две кнопки (Кнопка) с надписями: «блокировать», «разблокировать». Создайте обработчики события нажатия на кнопки, которые будут делать активным или неактивным поле ввода. Создайте обработчик события нажатия кнопки мыши на форме (Щелчок), который будет устанавливать цвет формы и делать невидимыми все элементы.

**Задание 14.** Реализуйте игру минер на поле 3×3 из кнопок (Кнопка). Первоначально все кнопки не содержат надписей. При попытке нажатия на кнопку на ней либо показывается количество мин, либо надпись «мина!» и меняется цвет окна.

**Задание 15.** Разместите на форме четыре кнопки (Кнопка). Напишите для каждой обработчик события, который будет менять размеры и местоположение на окне других кнопок.

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## **Практическое занятие № 6**

**Тема раздела:** Проектирование и отладка мобильных приложений

**Тема практического занятия:** Обработка событий: подсказки.

Обработка событий: цветовая индикация.

**Цель:** изучить обработку событий и цветовую индикацию.

**Планируемые результаты:**

**уметь:** обрабатывать события: подсказки и цветовую индикацию.

**знать:** обработку событий: подсказки и цветовая индикация.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Создать SDI-приложение (Single Document Interface, однодокументный интерфейс) с элементами ввода и отображения полей класса из задания к лабораторной работе 2. Для этого используйте различные элементы управления: текстовые поля, списки, независимые и радиокнопки, а также панели и менеджеры компоновки.

**Задание 2.** Ввод новых данных осуществлять через дополнительную диалоговую форму.

**Задание 3.** При изменении данных запрашивать подтверждение через окно диалога. В случае неполных данных сообщать об ошибке.

**Задание 4.** Объекты сохранять в коллекции.

**Задание 5.** Реализовать просмотр всей коллекции объектов через список. Для редактирования выбранного объекта создать дополнительную форму модального диалога.

**Задание 6.** Добавить на форму меню, позволяющее работать с пунктами: добавить, просмотреть, удалить, редактировать, справка.

**Задание 7.** Дублировать основные операции панелью инструментов.

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 7

**Тема раздела:** Проектирование и отладка мобильных приложений

**Тема практического занятия:** Подготовка стандартных модулей. Обработка событий: переключение между экранами.

**Цель:** Подготовка стандартных модулей. Обработка событий: переключение между экранами.

**Планируемые результаты:**

**уметь:** подготавливать стандартные модули и обрабатывать событие: переключение между экранами.

**знать:** стандартные модули и обработку событий: переключение между экранами.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Разработать мобильное приложение игры змейка. В программе должна быть панель с настройками, и главным меню. (создать не менее трех уровней)

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 8

**Тема раздела:** Проектирование и отладка мобильных приложений

**Тема практического занятия:** Передача данных между модулями.

**Цель:** ознакомиться с передачей данных между модулями.

**Планируемые результаты:**

**уметь:** передавать данные между модулями.

**знать:** передачу данных между модулями.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Передать текстовую строку из основной формы в дочернюю.

**Задание 2.** Передать данные из дочерней формы в основную.

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 9

**Тема раздела:** Проектирование и отладка мобильных приложений

**Тема практического занятия:** Разработка интерфейса для смартфонов.

Принцип юзабилити.

**Цель:** ознакомиться с разработкой интерфейсов для смартфонов и принципом юзабилити.

**Планируемые результаты:**

**уметь:** разрабатывать интерфейс для мобильных устройств. Изучить принцип юзабилити.

**знать:** разработку интерфейса для смартфонов. Принцип юзабилити.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

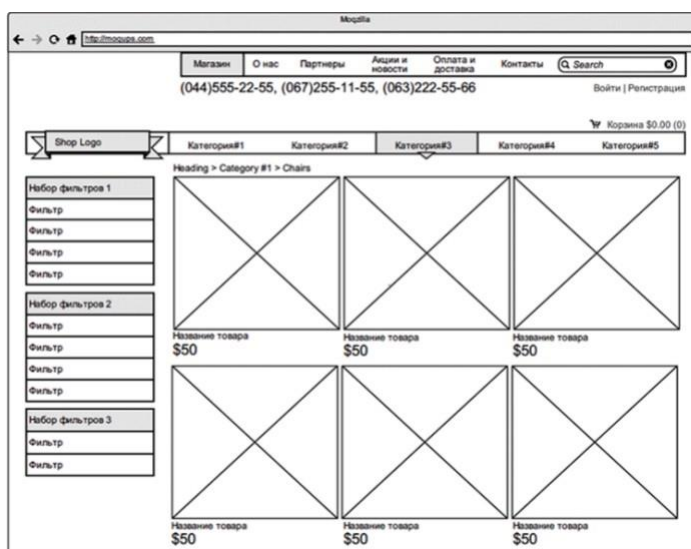
**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Разработать примерное мобильное приложение интернет-



магазина по образцу:

**Задание 2.** Разработать и спроектировать удобный интерфейс мобильного приложения доставки еды из ресторана.

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50



## Практическое занятие № 10

**Тема раздела:** Проектирование и отладка мобильных приложений

**Тема практического занятия:** Разметка активностей. компоновка UI-элементов на экране приложения

**Цель:** ознакомиться с разметкой активности и компоновкой UI – элементов на экране приложения.

**Планируемые результаты:**

**уметь:** работать с разметкой активности и компоновкой UI – элементов на экране приложения.

**знать:** разметку активности и компоновку UI – элементов на экране приложения.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Разработать мобильное приложение и выполнить разметку и UI компоновку.

**Задание 2.** Выполнить реализацию, компиляцию, отладку и тестирования Приложения.

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 11

**Тема раздела:** Проектирование и отладка мобильных приложений

**Тема практического занятия:** Тестирование и оптимизация мобильного приложения

**Цель:** ознакомиться с тестированием и оптимизацией мобильного приложения.

**Планируемые результаты:**

**уметь:** тестировать и оптимизировать мобильное приложение.

**знать:** тестирование и оптимизацию мобильного приложения.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Произведите тестирование и оптимизацию мобильной игры змейка сделанной в 7 практической работы.

**Задание 2.** В разработанное мобильное приложение про интернет-магазин из практической работы 9 выложите и придумайте 5 товаров. Все элементы интерфейса должны быть активны. Произвести оптимизацию и тестирование готового мобильного приложения.

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 12

**Тема раздела:** Проектирование и отладка мобильных приложений

**Тема практического занятия:** Основы создания мобильных приложений

**Цель:** ознакомиться с основами создания мобильных приложений.

**Планируемые результаты:**

**уметь:** создавать мобильные приложения.

**знать:** основы создания мобильных приложений.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

№ пп	Наименование	Цена, руб	Количество, шт	Дата привоза	Срок годности, лет
1	Шампунь	150	5	02.02.2019	3
2	Зубная паста	80.50	30	03.03.2017	4
3	Мыло	10.70	40	25.03.2013	4
4	Гель для душа	89.00	2	13.04.2019	2

**Задание 1.** Создать мобильное приложение в виде таблицы. В таблицу можно вводить данные о поступлении товара на производство.

**Задание 2.** Создать мобильное приложение игры «Doodle Jump».

**Задание 3.** Создать мобильное приложение игры «лабиринт»

**Задание 4.** Разработать мобильное приложение игры «Пазлы»

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

### Практическое занятие № 13

**Тема раздела:** Основы работы в ОС Android

**Тема практического занятия:** Архитектура платформы Android.

Уровень ядра. Уровень библиотек

**Цель:** ознакомиться с архитектурой платформы Android. Уровень ядра. Уровень библиотек

**Планируемые результаты:**

**уметь:** работать с архитектурой платформы Android. Уровень ядра. Уровень библиотек

**знать:** архитектуру платформы Android. Уровень ядра. Уровень библиотек

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Используя уровни библиотек разработать мобильное приложение «Новости»

**Задание 2.** Используя уровни библиотек создать мобильное приложение «Зумма»

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 14

**Тема раздела:** Основы работы в ОС Android

**Тема практического занятия:** Ресурсы в Android -приложениях

**Цель:** ознакомиться с ресурсами в Android -приложениях

**Планируемые результаты:**

**уметь:** работать с ресурсами в Android -приложениях

**знать:** ресурсы в Android -приложениях

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Используя ресурсы в Android -приложениях создать мобильное приложение «Счетчик калорий»

**Задание 2.** Используя ресурсы в Android -приложениях создать мобильное приложение «Счетчик денежных средств»

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 15

**Тема раздела:** Использование баз данных и развертывание мобильных приложений

**Тема практического занятия:** Расширения MS VisualStudio для разработки мобильных приложений

**Цель:** ознакомиться с расширениями MS VisualStudio для разработки мобильных приложений

**Планируемые результаты:**

**уметь:** работать с расширениями MS VisualStudio для разработки мобильных приложений

**знать:** расширения MS VisualStudio для разработки мобильных приложений

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Используя расширения MS VisualStudio создайте мобильное приложение игры «Шашки»

**Задание 2.** Используя расширения MS VisualStudio создайте мобильное приложение «Расписание»

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 16

**Тема раздела:** Использование баз данных и развертывание мобильных приложений

**Тема практического занятия:** Использование базы данных в мобильном приложении

**Цель:** ознакомиться с работой базы данных в мобильных приложениях.

**Планируемые результаты:**

**уметь:** работать с базой данных в мобильных приложениях.

**знать:** базу данных в мобильных приложениях.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Разработать мобильное приложение «Телефонный справочник» используя базу данных, где пользователь может добавлять и удалять телефонные номера.

**Задание 2.** В разработанном мобильном приложении в 9 практической работе «Интернет магазин» создать вкладку для регистрации пользователей для ввода логина и пароля. Данные должны храниться с использованием базы данных.

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## Практическое занятие № 17

**Тема раздела:** Использование баз данных и развертывание мобильных приложениях

**Тема практического занятия:** Дополнительные возможности при разработке мобильных приложений

**Цель:** ознакомиться с дополнительными возможностями при разработке мобильных приложений.

**Планируемые результаты:**

**уметь:** использовать дополнительные возможности при разработке мобильных приложений.

**знать:** дополнительные возможности при разработке мобильных приложений.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Разработать мобильное приложение для регистрации пользователей по примеру:



The image shows a mobile application interface for a registration form. The title 'feedback' is at the top. Below it are three input fields: 'Name' with the value 'Arturo Toledo', 'E-Mail' with the value 'arturot@microsoft.com', and a larger 'Comment' field. At the bottom right is a 'send' button.

**Задание 2.** Используя мобильное приложение Интернет-магазин разработанную в практической работе 9 создать панель покупки через банковскую карту нужного товара пользователю.



**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

**Практическое занятие № 18**

**Тема раздела:** Использование баз данных и развертывание мобильных приложениях

**Тема практического занятия:** Отладка и развертывание мобильного приложения

**Цель:** ознакомиться с отладкой и развертыванием мобильного приложения.

**Планируемые результаты:**

**уметь:** реализовывать отладку и развертывание мобильного приложения.

**знать:** отладку и развертывание мобильного приложения.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Создать мобильное приложение «Крестики-нолики» и произвести отладку мобильного приложения.

**Задание 2.** Создать мобильное приложение «Многофункциональный калькулятор» и произвести отладку мобильного приложения.

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69

## МДК. 01.04 Системное программирование

### Практическое занятие № 1

**Тема раздела:** Файловые системы и функций символьного ввода/вывода

**Тема практической работы:** Вывод на консоль сообщений и подсказок для пользователя

**Цель:** ознакомиться с. файловой системой и функциями символьного ввода/вывода

#### Планируемые результаты:

**уметь:** работать с файловой системой и функциями символьного ввода/вывода.

**знать:** понятия "файловая система", функции символьного ввода/вывода.

#### Порядок выполнения задания, методические указания:

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

#### Этапы выполнения работы:

**Задание 1.** Ознакомиться с теоретическими положениями

Книга: Системное программирование в среде Windows

#### Стандартные устройства и консольный ввод/вывод

Как и в UNIX, в Windows предусмотрены три стандартных устройства, предназначенные, соответственно, для ввода данных (input), вывода данных (output) и вывода сообщений об ошибках (error). В UNIX для этих устройств используются известные системе значения дескрипторов файлов (0, 1 и 2), однако в Windows доступ к стандартным устройствам осуществляется с помощью дескрипторов типа HANDLE, для получения которых предоставляется специальная функция.

`HANDLE GetStdHandle(DWORD nStdHandle)`

Возвращаемое значение: в случае успешного выполнения — действительный дескриптор, иначе — значение `INVALID_HANDLE_VALUE`.

Параметры

Параметр `nStdHandle` должен принимать одно из следующих значений:

- `STD_INPUT_HANDLE`
- `STD_OUTPUT_HANDLE`

- **STD\_ERROR\_HANDLE**

В качестве стандартных устройств обычно назначаются консоль и клавиатура. Стандартный ввод/вывод можно перенаправлять на другие устройства.

Вызов функции `GetStdHandle` не приводит к созданию новых или дублированию существующих дескрипторов стандартных устройств. Последовательные вызовы с указанием в качестве аргумента одного и того же устройства будут возвращать одно и то же значение дескриптора. Заккрытие дескриптора стандартного устройства делает это устройство недоступным для дальнейшего использования. По этой причине в примерах ниже мы будем часто открывать дескриптор стандартного устройства, но не закрывать его.

**BOOL SetStdHandle(DWORD nStdHandle, HANDLE hHandle)**

Возвращаемое значение: в случае успешного выполнения — **TRUE**, иначе — **FALSE**.

Параметры

Допустимые значения параметра `nStdHandle` функции `SetStdHandle` являются теми же, что и в случае функции `GetStdHandle`. Параметр `hHandle` указывает открытый файл, который назначается в качестве стандартного устройства.

Одним из методов перенаправления стандартного ввода/вывода является последовательный вызов функций `SetStdHandle` и `GetStdHandle`. Полученный в результате этого дескриптор используется в последующих операциях ввода/вывода.

Для указания путей доступа к консольному вводу (клавиатуре) и консольному выводу предусмотрены два зарезервированных имени: **"CONIN\$"** и **"CONOUT\$"**. Роль стандартных устройств ввода, вывода и вывода ошибок первоначально отводится консоли. Однако консоль можно использовать даже после того, как операции ввода/вывода, требующие применения стандартных устройств, будут перенаправлены; для этого требуется лишь открыть дескрипторы для файлов **"CONIN\$"** и **"CONOUT\$"**, вызвав функцию `CreateFile`.

Первый метод является косвенным и основывается на том, что функция `dup` возвращает дескриптор файла с наименьшим доступным номером. Предположим, вы хотите переназначить стандартный ввод (файловый дескриптор 0) открытому файлу, описанному как `fd_redirect`. Тогда можно записать следующий код:

```
close (STDIN_FILENO);  
dup (fd_redirect);
```

Во втором методе используется функция `dup2`, а третий метод предполагает вызов замысловатой перегруженной функции `fcntl` с использованием в качестве параметра значения `F_DUPFD`.

Операции консольного ввода/вывода могут выполняться с помощью функций `ReadFile` и `WriteFile`, но проще использовать предназначенные специально для этого функции консоли `ReadConsole` и `WriteConsole`.

Основное преимущество этих функций заключается в том, что они манипулируют не байтами, а обобщенными символами (TCHAR), и, кроме того, обрабатывают символы в соответствии с текущими режимами консоли, которые устанавливаются функцией SetConsoleMode.

BOOL SetConsoleMode(HANDLE hConsoleHandle, DWORD fdevMode)

Возвращаемое значение: тогда, и только тогда, когда функция завершается успешно — TRUE, иначе — FALSE.

Параметры

hConsoleHandle — дескриптор буфера ввода консоли или буфера дисплея, который должен быть создан с правами доступа GENERIC\_WRITE, даже если устройство предназначено только для ввода информации.

Параметр fdevMode задает способ обработки символов. В имени каждого из его флагов содержится компонент, указывающий, к чему относится данный флаг — к вводу (input) или выводу (output). Ниже перечислены пять обычно используемых флагов, причем все они устанавливаются по умолчанию.

- ENABLE\_LINE\_INPUT — возврат из функции чтения (ReadConsole) происходит только после считывания символа возврата каретки.

- ENABLE\_ECHO\_INPUT — эхо-отображение вводимых символов на экране.

- ENABLE\_PROCESSED\_INPUT — установка этого флага приводит к обработке системой управляющих символов возврата на одну позицию, возврата каретки и перехода на новую строку.

- ENABLE\_PROCESSED\_OUTPUT — установка этого флага приводит к обработке системой управляющих символов возврата на одну позицию, табуляции, подачи звукового сигнала, возврата каретки и перехода на новую строку.

- ENABLE\_WRAP\_AT\_EOL\_OUTPUT — переход на следующую строку экрана как при обычном выводе символов, так и при их эхо-отображении в процессе ввода.

В случае неудачного завершения функции SetConsoleMode текущий режим остается неизменным, и функция возвращает значение FALSE. Как обычно, для получения номера ошибки следует воспользоваться функцией GetLastError.

Функции ReadConsole и WriteConsole аналогичны функциям ReadFile и WriteFile.

BOOL ReadConsole(HANDLE hConsoleInput, LPVOID lpBuffer, DWORD cchToRead, LPDWORD lpCchRead, LPVOID lpReserved)

Возвращаемое значение: тогда, и только тогда, когда функция завершается успешно — TRUE, иначе — FALSE.

Параметры у этой функции почти те же, что и у функции ReadFile. Значения обоих параметров, связанных с количеством подлежащих считыванию (cchToRead) и фактически считанных (lpCchRead) символов, выражаются в терминах обобщенных символов, а не байтов, а значение параметра lpReserved должно быть равным NULL. Как и во всех остальных

подобных случаях, никогда не используйте для собственных нужд зарезервированные поля, аналогичные `IpReserved`, которые встречаются в некоторых функциях. Параметры функции `WriteConsole` имеют тот же смысл и не нуждаются в дополнительных пояснениях. В очередном примере будет проиллюстрировано применение функций `Read-Console` и `WriteConsole`, и, кроме того, будет показано, как использовать возможности управления режимом консоли.

Любому процессу в каждый момент времени может быть назначена только одна консоль. Приложениям того типа, с которым мы имели дело до сих пор, консоль передается обычно на стадии инициализации. Однако в целом ряде других случаев, например, в случае серверных или GUI-приложений, у вас может возникнуть необходимость в получении отдельной консоли, на которую можно было бы выводить информацию о состоянии программы или отладочную информацию. Для этих целей можно воспользоваться двумя простыми функциями, не имеющими параметров.

`BOOL FreeConsole(VOID)`

`BOOL AllocConsole(VOID)`

Функция `FreeConsole` отключает процесс от его консоли, тогда как функция `AllocConsole` создает новую консоль, ассоциированную с дескрипторами стандартного ввода информации, стандартного вывода информации и стандартного вывода сообщений об ошибках, принадлежащими данному процессу. Если консоль у процесса уже имеется, функция `AllocConsole` завершится с ошибкой; чтобы избежать этого, следует предварительно вызывать функцию `FreeConsole`.

**Примечание**

GUI-приложения Windows не имеют консоли по умолчанию и должны получить ее, прежде чем смогут воспользоваться функциями `WriteConsole` или `printf` для вывода на консоль. Процессы на стороне сервера также могут не иметь консоли. О том, как создать процесс без консоли, рассказано в главе 6.

Имеется также множество других функций консольного ввода/вывода, предназначенных для установки позиции курсора, а также задания атрибутов выводимых символов (например, цвета) и так далее. Принятый в данной книге подход состоит в том, чтобы использовать лишь те функции, которые необходимы для создания примеров работоспособных программ, поэтому углубляться больше, чем это необходимо, в пользовательские интерфейсы мы не будем. После того как вы разберете примеры, для вас не составит большого труда изучить дополнительные функции, воспользовавшись справочными материалами.

**Пример: вывод на консоль сообщений и подсказок для пользователя**

Функция `ConsolePrompt`, входящая в программу 1, является полезной утилитой, которая выводит на консоль заданное сообщение и возвращает ответ пользователя на него. Данная утилита предусматривает возможность

подавления эхо-отображения ответной информации, полученной от пользователя. В указанной функции используются функции консольного ввода/вывода и обобщенные символы. Двумя другими точками входа в этом модуле являются функции Print-Strings и PrintMsg; эти функции допускают использование любого дескриптора, однако обычно они применяются совместно с дескрипторами устройств стандартного вывода информации и стандартного вывода сообщений об ошибках. В первой функции разрешается использовать список аргументов переменной длины, тогда как во второй в качестве аргумента можно задавать только одну строку, что в некоторых случаях может оказаться удобнее. Для обработки списка аргументов переменной длины функция PrintStrings использует функции `va_start`, `va_arg` и `va_end` стандартной библиотеки C.

Описанные функции, а также функции из обобщенной библиотеки C будут привлекаться для использования в приводимых в данной книге примерах программ при всякой удобной возможности.

#### Примечание

Коды программ, находящиеся на Web-сайте книги, содержат подробные комментарии и тщательно документированы, тогда как в самой книге большинство комментариев с целью сокращения места были опущены, и основное внимание в ней уделяется использованию Windows.

Следует также отметить, что в примере вводится заголовочный файл `Envirmnt.h` (его код приведен в приложении A и предоставлен на Web-сайте книги), который должен использоваться совместно со всеми приводимыми в книге программами. Этот файл содержит определения символических констант `UNICODE` и `_UNICODE` (сами определения "закомментированы"; при компоновке приложений, предназначенных для работы с символами стандарта Unicode, символы комментариев следует удалить), а также необходимых макропеременных, учитывающих особенности окружения. В заголовочных файлах, находящихся на Web-сайте, определены также дополнительные модификаторы, которые обеспечивают импорт и экспорт имен функций, а также гарантируют соблюдение соответствующих соглашений о вызове функций.

Программа 1. PrintMsg: вспомогательные функции вывода на консоль сообщений и ожидания ответа от пользователя

```
/* PrintMsg.c: ConsolePrompt, PrintStrings, PrintMsg */
#include "Envirmnt.h" /* В этом файле устанавливаются директивы #define и
#include <windows.h>
#include <stdarg.h>
```

```

BOOL PrintStrings (HANDLE hOut, ...)
/* Запись сообщений в буфер экрана консоли. */
{
    DWORD MsgLen, Count;
    LPCTSTR pMsg;
    va_list pMsgList; /* Строка текущего сообщения. */
    va_start (pMsgList, hOut); /* Начать обработку сообщений. */
    while ((pMsg = va_arg(pMsgList, LPCTSTR)) != NULL) {
        MsgLen = _tcslen(pMsg);
        /* Функция WriteConsole может применяться только с дескриптором буфера
        экрана консоли. */
        if (!WriteConsole(hOut, pMsg, MsgLen, &Count, NULL))
            /* Функция WriteFile вызывается только в случае неудачного завершения
            функции WriteConsole. */
            && !WriteFile(hOut, pMsg, MsgLen * sizeof (TCHAR), &Count, NULL))
        return FALSE;
    }
    va_end(pMsgList);
    return TRUE;
}

BOOL PrintMsg(HANDLE hOut, LPCTSTR pMsg)
/* Версия PrintStrings для вывода одиночного сообщения. */
{
    return PrintStrings(hOut, pMsg, NULL);
}

BOOL ConsolePrompt(LPCTSTR pPromptMsg, LPTSTR pResponse, DWORD
MaxTchar, BOOL Echo)
/* Вывести на консоль подсказку для пользователя и получить от него ответ.
*/
{
    HANDLE hStdIn, hStdOut;
    DWORD TcharIn, EchoFlag;
    BOOL Success;
    hStdIn = CreateFile(_T("CONIN$"), GENERIC_READ | GENERIC_WRITE, 0,
    NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    hStdOut = CreateFile(_T("CONOUT$"), GENERIC_WRITE, 0, NULL,
    OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    EchoFlag = Echo ? ENABLE_ECHO_INPUT : 0;
    Success = SetConsoleMode(hStdIn, ENABLE_LINE_INPUT | EchoFlag |

```



```

ENABLE_PROCESSED_INPUT) &&
    SetConsoleMode (hStdOut, ENABLE_WRAP_AT_EOL_OUTPUT |
ENABLE_PROCESSED_OUTPUT) &&
    PrintStrings (hStdOut, pPromptMsg, NULL) &&
    ReadConsole (hStdIn, pResponse, MaxTchar, &TcharIn, NULL);
if (Success) pResponse [TcharIn - 2] = ";
CloseHandle (hStdIn);
CloseHandle (hStdOut);
return Success;
}

```

Обратите внимание, что при вычислении возвращаемого функцией значения булевской переменной `Success`, которое служит индикатором успешности выполнения, в программе, с выгодой для логики ее работы, используется тот факт, что стандартом ANSI C гарантируется так называемое "сокращенное" вычисление логических выражений в направлении слева направо; поэтому, как только при вычислении части выражения, расположенной слева от любой из операций логического "и" (&&), в качестве результата будет получено значение `FALSE`, оставшая часть выражения, расположенная справа от данной операции, вычисляться не будет, поскольку результат вычисления всего выражения в целом оказывается предопределенным. Данный стиль написания программ может показаться чересчур компактным, однако он обладает тем преимуществом, что позволяет организовать логически стройную и понятную последовательность системных вызовов, не загромождая программу многочисленными операторами условных переходов. Для получения более подробной информации о возможных ошибках можно воспользоваться функцией `GetLastError`. Распространенный в Windows возврат функциями логических значений поощряет подобную практику.

В данной функции сообщения об ошибках не выводятся; их вывод, если это будет необходимо, можно предусмотреть в вызывающей программе.

В программном коде используется тот документированный факт, что при использовании функции `WriteConsole` вместе с дескриптором, который не является дескриптором консоли, ее выполнение будет завершено с ошибкой. В связи с этим предварительный запрос свойств дескриптора не является обязательным. Функция воспользуется консольным режимом лишь в том случае, если указанный в ее вызове дескриптор действительно связан с консолью.

Кроме того, функция `ReadConsole` возвращает управляющие символы возврата каретки и перехода на новую строку, что диктует необходимость

вставки дополнительных нулевых символов после символов возврата каретки в соответствующих местах.

**Задание 2.** Воспроизведите текст программы. Выполните его. Внесите изменения так, чтобы на выходе получить следующие данные: ФИО (свои), номер группы и результаты сдачи трех экзаменов.

**Задание 3.** Подготовить отчет о выполненной работе

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## **Практическое занятие № 2**

**Тема раздела:** Файловые системы и функции символьного ввода/вывода

**Тема практической работы:** Обработка ошибок

**Цель:** ознакомиться с обработкой ошибок.

**Планируемые результаты:**

**уметь:** анализировать и устранять системные ошибки .

**знать:** понятие "системная ошибка".

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Ознакомиться с теоретическими положениями

Книга: [Системное программирование в среде Windows](#)

Пример: обработка ошибок

Функция FormatMessage превращает простой номер сообщения в описательное сообщение, представляющее собой фразу на английском или любом другом из множества возможных языков, и возвращает размер сообщения.

В программе 2 представлена полезная универсальная функция `ReportError`, предназначенная для обработки ошибок и по своим возможностям аналогичная входящей в состав библиотеки C функции `report`, а также функциям `err_sys` и `err_ret`. Функция `ReportError` передает в выходной буфер сообщение в виде, определяемом первым аргументом, и либо прекращает выполнение с кодом выхода по ошибке, либо осуществляет обычный возврат управления, в зависимости от значения второго аргумента. Третий аргумент определяет, должны ли отображаться системные сообщения об ошибках.

Обратите внимание на аргументы функции `FormatMessage`. В качестве одного из параметров используется значение, возвращаемое функцией `GetLastError`, а на необходимость генерации сообщения системой указывает флаг. Сгенерированное сообщение сохраняется в буфере, выделяемом функцией, а соответствующий адрес возвращается в параметре. Имеются также другие параметры, для которых указаны значения по умолчанию. Язык сообщений может быть задан как во время компиляции, так и во время выполнения. В этой книге функция `Format-Message` далее нигде не используется, поэтому никаких дополнительных пояснений относительно нее в тексте не дается.

Функция `ReportError` упрощает обработку ошибок, и будет использоваться почти во всех последующих примерах.

В программе 2 вводится заголовочный файл `EvryThng.h`. Как следует из самого его названия, этот файл включает в себя файлы `<windows.h>`, `Envirmnt.h` и все остальные заголовочные файлы, которые были явно указаны в программе 1. Кроме того, в нем описаны такие обычно используемые функции, как `PrintMsg`, `PrintStrings` и `ReportError`. Во всех остальных примерах будет использоваться только этот заголовочный файл, листинг которого приведен в приложении А.

Программа 2. Функция `Report Error`, предназначенная для вывода сообщений об ошибках при выполнении системных вызовов

```
#include "EvryThng.h"
VOID ReportError(LPCTSTR UserMessage, DWORD ExitCode, BOOL
PrintErrorMsg)
/* Универсальная функция для вывода сообщений о системных ошибках. */
{
    DWORD eMsgLen, LastErr = GetLastError();
    LPTSTR lpvSysMsg;
    HANDLE hStdErr = GetStdHandle(STD_ERROR_HANDLE);
    PrintMsg(hStdErr, UserMessage);
    if (PrintErrorMsg) {
        eMsgLen = FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
FORMAT_MESSAGE_FROM_SYSTEM, NULL, LastErr,
MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
(LPTSTR)&lpvSysMsg, 0, NULL);
        PrintStrings (hStdErr, _T("\n"), lpvSysMsg, _T("\n"), NULL);
    }
}
```

```

/* Освободить блок памяти, содержащий сообщение об ошибке. */
HeapFree(GetProcessHeap(), 0, lpvSysMsg); /* См. гл. 5. */
}
if (ExitCode > 0) ExitProcess (ExitCode);
else return;
}

```

## ПРИЛОЖЕНИЕ А

### Использование примеров программ

На Web-сайте книги (<http://www.awprofessional.com/titles/0321256190>) находится zip-архив, который содержит исходные тексты всех примеров программ, а также соответствующие заголовочные файлы, служебные функции, файлы проектов и исполняемые файлы. Ряд программ демонстрируют дополнительные возможности и предоставляют решения отдельных упражнений, однако на Web-сайте приведены решения не для всех упражнений и представлены не все из упоминающихся в книге альтернативных вариантов реализации программ.

- Все программы тестировались под управлением Windows 2000, XP и Server 2003 на самых различных системах, от лэптопов до серверов. В необходимых случаях тестирование осуществлялось под управлением Windows 9x, хотя многие программы — особенно те, которые предлагаются на более поздних этапах изложения материала — под управлением Windows 9x и даже NT 4.0 выполняться не будут.

- Сборка и выполнение программ осуществлялись как с включенными определениями UNICODE, так и без таковых. Под управлением Windows 9x будут работать лишь те программы, в которых возможность работы с символами в кодировке UNICODE не предусмотрена.

- В подавляющем большинстве случаев компиляция программ в интегрированной среде разработки Microsoft Visual C++ версий 7.0 и 6.0 не будет сопровождаться выдачей предупреждающих сообщений, если для критерия серьезности ошибок (warning level), которые должны сопровождаться выводом диагностических сообщений компилятора, установлено значение 3. Однако существуют некоторые незначительные исключения, например, вывод сообщения "Отсутствует оператор return в основной программе" ("no return from main program") в случае использования функции ExitProcess.

- Для проектов Microsoft Visual Studio .NET и Microsoft Visual Studio C++ 6.0 предусмотрены разные каталоги, каковыми являются каталоги Projects7 и Projects6. Соответствующие исполняемые файлы программ помещаются в каталоги run7 и run6.

- В программах широко применяются функции обобщенной библиотеки C, а также такие специфические для используемых типов компиляторов ключевые слова, как \_\_try, \_\_except или \_\_leave. Начиная с главы 7, важную роль в программах играют многопоточная библиотека C времени выполнения и функции \_beginthreadex и \_endthreadex.

- Предоставляются как файлы проектов (в их окончательной (release), а не отладочной (debug) форме), так и make-файлы. Все проекты достаточно просты, характеризуются минимальным количеством зависимостей (dependencies) и их можно быстро создать заново в любой желаемой конфигурации с получением либо отладочной, либо окончательной версии.

- Проекты для построения всех программ, за исключением статических и динамических библиотек, ориентированы на создание консольных приложений.

Для сборки программ можно воспользоваться также такими инструментальными средствами, распространяемыми в рамках проекта программного обеспечения с открытым исходным кодом (GNU), как компиляторы gcc и g++, входящие в состав комплекта инструментов Gnu Compiler Collection (<http://gcc.gnu.org/>). Читатели, заинтересованные в подобных средствах разработки, должны ознакомиться с действующим на условиях GNU проектом MinGW (<http://www.mingw.org>), который описывается как "совокупность свободно доступных и свободно распространяемых заголовочных файлов и библиотек импорта, специфических для Windows, объединенных с наборами инструментов GNU, что позволяет создавать программы для среды Windows, не зависящие от динамических библиотек C времени выполнения, выпускаемых третьими сторонами". В то же время, при тестировании большинства примеров программ, приведенных в книге, я эти средства не применял, но весьма успешно использовал возможности MinGW, и мне даже удавалось выполнять межплатформенную сборку для создания исполняемых программ и DLL-библиотек Windows в Linux-системах. Более того, я имел возможность убедиться в чрезвычайно высокой эффективности систем диагностики ошибок и вывода предупреждающих сообщений компиляторов gcc и g++ при разработке 64-разрядных программ.

**Задание 2.** Воспроизведите текст программы. Выполните его.

Сформировать отчет об ошибках.

**Задание 3.** Подготовить отчет о выполненной работе

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

### Практическое занятие № 3

**Тема раздела:** Файловые системы и функции символьного ввода/вывода

**Тема практической работы:** Копирование нескольких файлов на стандартное устройство вывода

**Цель:** ознакомиться с приемами копирования нескольких файлов на стандартное устройство вывода.

**Планируемые результаты:**

**уметь:** Копировать группы файлов на стандартное устройство вывода.

**знать:** понятие "маска", приемы копирования.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Ознакомиться с теоретическими положениями

Книга: Системное программирование в среде Windows

Пример: копирование нескольких файлов на стандартное устройство вывода

В программе 3 иллюстрируется использование стандартных устройств ввода/вывода, а также демонстрируется, как улучшить контроль ошибок и усовершенствовать взаимодействие с пользователем. Эта программа представляет собой вариант ограниченной реализации команды UNIX `cat`, которая копирует один или несколько заданных файлов (или содержимое буфера стандартного устройства ввода, если файлы не указаны) на стандартное устройство вывода.

Программа 3 включает полную обработку ошибок. В большинстве других примеров проверка ошибок опущена или сведена к минимуму, но полностью включена в завершенные документированные варианты программ, находящиеся на Web-сайте. Обратите внимание на функцию `Options` (ее листинг приведен в приложении А), вызываемую в начале программы. Эта функция, которая включена в состав программ, находящихся на Web-сайте, и используется на протяжении всей книги, просматривает параметры в командной строке и возвращает индекс массива `argv`, соответствующий имени первого файла. Функция `Options` аналогична функции `getopt`, которая используется во многих программах в UNIX.

Программа 3. `cat`: вывод нескольких файлов на стандартное устройство вывода

/\* Глава 2. `cat`. \*/

/\* `cat` [параметры] [файлы] Допускается только параметр `-s`,

```

предназначенный для подавления вывода сообщений об ошибках в случае,
если один из указанных файлов не существует. */
#include "EvryThng.h"
#define BUF_SIZE 0x200
static VOID CatFile(HANDLE, HANDLE);
int _tmain(int argc, LPTSTR argv[]) {
    HANDLE hInFile, hStdIn = GetStdHandle(STD_INPUT_HANDLE);
    HANDLE hStdOut = GetStdHandle(STD_OUTPUT_HANDLE);
    BOOL DashS;
    int iArg, iFirstFile;
    /* Переменная DashS будет установлена только в случае задания параметра
    "-s" в командной строке. */
    /* iFirstFile — индекс первого входного файла в списке argv[]. */
    iFirstFile = Options(argc, argv, _T("s"), &DashS, NULL);
    if (iFirstFile == argc) { /*Отсутствие входных файлов в списке аргументов.*/
        /* Использовать стандартное устройство ввода. */
        CatFile(hStdIn, hStdOut);
        return 0;
    }
    /* Обработать каждый входной файл. */
    for (iArg = iFirstFile; iArg < argc; iArg++) {
        hInFile = CreateFile(argv[iArg], GENERIC_READ, 0, NULL,
        OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);
        if (hInFile == INVALID_HANDLE_VALUE && !DashS) ReportError (_T("Cat:
ошибка при открытии файла"), 1, TRUE);
        CatFile(hInFile, hStdOut);
        CloseHandle(hInFile);
    }
    return 0;
}
/* Функция, выполняющая всю работу:
/* читает входные данные и копирует их на стандартное устройства вывода.
*/
static VOID CatFile(HANDLE hInFile, HANDLE hOutFile) {
    DWORD nIn, nOut;
    BYTE Buffer[BUF_SIZE];
    while (ReadFile(hInFile, Buffer, BUF_SIZE, &nIn, NULL) && (nIn != 0) &&
    WriteFile(hOutFile, Buffer, nIn, &nOut, NULL));
    return;
}

```

**Задание 2.** На основе списка файлов составить возможные маски.

Воспроизведите текст программы. Выполните его. Внесите изменения так, чтобы выполнить копирование файлов с использованием составленных масок

**Задание 3.** Подготовить отчет о выполненной работе

### Критерии оценивания:

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

### Практическое занятие № 4

**Тема раздела:** Файловые системы и функции символьного ввода/вывода

**Тема практической работы:** Печать текущего каталога

**Цель:** ознакомиться печатью текущего каталога.

**Планируемые результаты:**

**уметь:** обеспечивать взаимодействие ОС и печатающего устройства.

**знать:** особенности организации вывода данных на печать.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Ознакомиться с теоретическими положениями

#### Управление каталогами

Создание и удаление каталогов осуществляется при помощи двух простых функций.

BOOL CreateDirectory(LPCTSTR lpPathName,  
LPSECURITY\_ATTRIBUTES lpSecurityAttributes)  
BOOL RemoveDirectory(LPCTSTR lpPathName)

lpPathName является указателем на завершающуюся нулевым символом строку, которая содержит путь к создаваемому или удаляемому каталогу. Как и в случае других функций, на данном этапе атрибуты защиты файла должны полагаться равными NULL. Удалить можно только пустой каталог.



Как и в UNIX, у каждого процесса имеется текущий, или рабочий, каталог. Кроме того, для каждого диска поддерживается свой рабочий каталог. Программист может как устанавливать рабочий каталог, так и получать информацию о том, какой каталог в данный момент является текущим. Первая функция предназначена для установки каталогов.

**BOOL SetCurrentDirectory(LPCTSTR lpPathName)**

lpPathName определяет путь к новому текущему каталогу. Это может быть относительный путь или абсолютный полный путь, в начале которого указаны либо буква диска и двоеточие (например, D:), либо имя UNC (например, ACCTG\_SERVERPUBLIC).

Если в качестве пути к каталогу указывается только имя диска (например, A: или C:), то рабочим каталогом становится рабочий каталог данного диска. Например, если рабочие каталоги устанавливались в последовательности:

```
C:MSDEV  
INCLUDE  
A:MEMOSTODO  
C:
```

то результирующим рабочим каталогом будет:

```
C:MSDEVINCLUDE
```

Следующая функция возвращает абсолютный полный путь к текущему каталогу, помещая его в буфер, предоставляемый программистом:

**DWORD GetCurrentDirectory(DWORD cchCurDir, LPTSTR lpCurDir)**

Возвращаемое значение: длина строки, содержащей путь доступа к текущему каталогу, или требуемый размер буфера, если буфер не достаточно велик; в случае ошибки — нуль.

cchCurDir — размер буфера, содержащего имя каталога, который определяется количеством символов (а не байт). Размер буфера должен рассчитываться с учетом завершающего нулевого символа строки. lpCurDir является указателем на буфер, предназначенный для получения строки, содержащей путь.

Заметьте, что в случае если размер буфера оказался недостаточным для того, чтобы в нем уместилась вся строка пути, функция возвратит значение, указывающее на требуемый размер буфера. Поэтому при тестировании успешности выполнения функции следует проверять два условия: равно ли возвращаемое значение нулю и не превышает ли оно значение, заданное аргументом cchCurDir.

Подобный метод возврата строк и их длины широко распространен в Windows и требует внимательной обработки результатов. Программа 6 иллюстрирует типичный фрагмент кода, реализующего эту логику. Аналогичная логика реализуется и в других примерах. Вместе с тем, указанный метод применяется не всегда. Некоторые функции возвращают булевские значения, а параметр размера в них используется дважды: перед вызовом функции его значение устанавливается равным размеру буфера, а

затем изменяется функцией. В качестве одного из многих возможных примеров можно привести функцию LookupAccountName,

### **Пример: печать текущего каталога**

Программа 6 реализует очередную версию команды UNIX pwd. Размер буфера определяется значением параметра MAX\_PATH, однако проверка ошибок все равно предусмотрена, чтобы проиллюстрировать работу функции GetCurrentDirectory.

Программа 6. pwd: печать текущего каталога

/\* Глава 2. pwd – вывод на печать содержимого рабочего каталога. \*/

```
#include "EvryThng.h"
```

```
#define DIRNAME_LEN MAX_PATH + 2
```

```
int _tmain(int argc, LPTSTR argv[]) {
```

```
    TCHAR pwdBuffer [DIRNAME_LEN];
```

```
    DWORD LenCurDir;
```

```
    LenCurDir = GetCurrentDirectory(DIRNAME_LEN, pwdBuffer);
```

```
    if (LenCurDir == 0) ReportError(_T("Не удастся получить путь."), 1, TRUE);
```

```
    if (LenCurDir > DIRNAME_LEN) ReportError(_T("Слишком длинный путь."),  
2, FALSE);
```

```
    PrintMsg(GetStdHandle(STD_OUTPUT_HANDLE), pwdBuffer);
```

```
    return 0;
```

```
}
```

**Задание 2.** Воспроизведите текст программы. Выполнить печать данных.

**Задание 3.** Подготовить отчет о выполненной работе

### **Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## **Практическое занятие № 5**

**Тема раздела:** Реестр

**Тема практической работы:** Вывод списка разделов и содержимого реестра

**Цель:** вывести список разделов и содержимое реестра.

**Планируемые результаты:**

**уметь:** работать с Реестром.

**знать:** Реестр.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Ознакомиться с теоретическими положениями

Книга: Системное программирование в среде Windows

**Реестр**

Реестр — это централизованная иерархическая база данных, хранящая информацию о параметрах конфигурации операционной системы и установленных приложений. Доступ к реестру осуществляется через разделы, или ключи, реестра (registry keys), играющие ту же роль, что и каталоги в файловой системе. Раздел может содержать подразделы или пары "имя-значение", в которых между именем и значением существует примерно та же взаимосвязь, что и между именами файлов и их содержимым.

Пользователь или системный администратор может просматривать и изменять содержимое реестра, пользуясь редактором реестра, для запуска которого необходимо выполнить команду REGEDIT. Реестром можно управлять также из программ, используя функции API реестра, описанные в данном разделе.

Справа на этом рисунке можно видеть специфическая информация, относящаяся к установленному на данном локальном компьютере процессору. В нижней левой части рисунка показаны различные разделы, содержащие информацию об установленном в локальной системе программном обеспечении. Обратите внимание, что каждый ключ обязательно имеет значение по умолчанию, которое указывается в списке самым первым, предшествуя любым другим парам "имя-значение".

Рассмотрение принципов реализации реестра, включая организацию хранения и извлечения хранящихся в реестре данных, выходит за рамки данной книги; для более глубокого изучения этих вопросов обратитесь к списку дополнительной литературы, приведенному в конце главы.

**Ключи реестра**

На рисунке показана аналогия между разделами реестра и каталогами файловой системы. Каждый раздел может содержать другие разделы или последовательности пар "имя-значение". В то время как доступ к файловой системе реализуется посредством указания путей доступа, доступ к реестру

осуществляется через его разделы. Существует несколько предопределенных разделов, которые играют роль точек входа в реестр.



Рис. Окно редактора реестра

- **HKEY\_LOCAL\_MACHINE.** В этом разделе хранится информация об оборудовании локального компьютера и установленном на нем программном обеспечении. Информация об установленном программном обеспечении обычно создается в подразделах (subkeys) в виде: SOFTWARE\НазваниеКомпании\НазваниеПродукта\Версия.

- **HKEY\_USERS.** В этом разделе хранится информация о настройке пользовательских конфигураций.

- **HKEY\_CURRENT\_CONFIG.** В этом разделе хранятся текущие настройки таких параметров, как разрешение дисплея или гарнитура шрифта.

- **HKEY\_CLASSES\_ROOT.** В этом разделе содержатся подчиненные записи, устанавливающие соответствие между именами файлов и классами, а также приложениями, используемыми оболочкой для доступа к объектам, имена которых имеют определенные расширения. В этот раздел также входят все подразделы, необходимые для функционирования модели компонентных объектов (Component Object Model — COM), разработанной компанией Microsoft.

- **HKEY\_CURRENT\_USER.** В этом разделе хранится информация, определяемая пользователем, в том числе информация о переменных среды, принтерах и предпочтительных для вошедшего в систему пользователя конфигурационных параметрах приложений.

### **Пример: вывод списка разделов и содержимого реестра**

Программа lsReq (программа 4), является видоизменением lsW (программа 3.2, предназначенная для вывода списка файлов и каталогов) и обрабатывает не каталоги и файлы, а разделы и пары "имя-значение" системного реестра.

Программа 4. lsReq: вывод списка разделов и содержимого системного реестра

```

/* Глава 3. IsReg: Команда вывода содержимого реестра. Адаптированная
версия программы 3.2. */
/* IsReg [параметры] подраздел */
#include "EvryThng.h"
BOOL TraverseRegistry(HKEY, LPTSTR, LPTSTR, LPBOOL);
BOOL DisplayPair(LPTSTR, DWORD, LPBYTE, DWORD, LPBOOL);
BOOL DisplaySubKey (LPTSTR, LPTSTR, PFILETIME, LPBOOL);
int _tmain(int argc, LPTSTR argv[]) {
    BOOL Flags[2], ok = TRUE;
    TCHAR KeyName[MAX_PATH + 1];
    LPTSTR pScan;
    DWORD i, KeyIndex;
    HKEY hKey, hNextKey;
    /* Таблица предопределенных имен и дескрипторов разделов. */
    LPTSTR PreDefKeyNames[] = {
        _T("HKEY_LOCAL_MACHINE"), _T("HKEY_CLASSES_ROOT"),
        _T("HKEY_CURRENT_USER"), _T("HKEY_CURRENT_CONFIG"), NULL
    };
    HKEY PreDefKeys[] = {
        HKEY_LOCAL_MACHINE, HKEY_CLASSES_ROOT,
        HKEY_CURRENT_USER, HKEY_CURRENT_CONFIG
    };
    KeyIndex = Options(argc, argv, _T("R"), &Flags[0], &Flags[1], NULL);
    /* "Разобрать" шаблон поиска на "раздел" и "подраздел". */
    /* Воссоздать раздел. */
    pScan = argv[KeyIndex];
    for (i = 0; *pScan != _T("\\") && *pScan != _T('\0'); pScan++, i++) KeyName[i] =
        *pScan;
    KeyName[i] = _T('\0');
    if (*pScan == _T("\\")) pScan++;
    /* Преобразовать предопределенное имя раздела в соответствующий
HKEY.*/
    for (i = 0; PreDefKeyNames[i] != NULL && _tcscmp(PreDefKeyNames[i],
KeyName) != 0; i++);
    hKey = PreDefKeys[i];
    RegOpenKeyEx(hKey, pScan, 0, KEY_READ, &hNextKey);
    hKey = hNextKey;
    ok = TraverseRegistry(hKey, argv[KeyIndex], NULL, Flags);
    return ok ? 0 : 1;
}
BOOL TraverseRegistry(HKEY hKey, LPTSTR FullKeyName, LPTSTR SubKey,
LPBOOL Flags)
/*Совершить обход разделов и подразделов реестра, если задан параметр –
R.*/
{

```

```

HKEY hSubK;
BOOL Recursive = Flags[0];
LONG Result;
DWORD ValType, Index, NumSubKs, SubKNameLen, ValNameLen, ValLen;
DWORD MaxSubKLen, NumVals, MaxValNameLen, MaxValLen;
FILETIME LastWriteTime;
LPTSTR SubKName, ValName;
LPBYTE Val;
TCHAR FullSubKName[MAX_PATH + 1];
/* Открыть дескриптор раздела. */
RegOpenKeyEx(hKey, SubKey, 0, KEY_READ, &hSubK);
/* Определить максимальный размер информации относительно раздела и
распределить память. */
RegQueryInfoKey(hSubK, NULL, NULL, NULL, &NumSubKs,
&MaxSubKLen, NULL, &NumVals, &MaxValNameLen, &MaxValLen, NULL,
&LastWriteTime);
SubKName = malloc (MaxSubKLen+1); /* Размер без учета завершающего
нулевого символа. */
ValName = malloc(MaxValNameLen+1); /* Учет нулевой символ. */
Val = malloc(MaxValLen); /* Размер в байтах. */
/* Первый проход: пары "имя-значение". */
for (Index = 0; Index < NumVals; Index++) {
    ValNameLen = MaxValNameLen + 1; /* Устанавливается каждый раз! */
    ValLen = MaxValLen + 1;
    RegEnumValue(hSubK, Index, ValName, &ValNameLen, NULL, &ValType,
Val, &ValLen);
    DisplayPair(ValName, ValType, Val, ValLen, Flags);
}
/* Второй проход: подразделы. */
for (Index = 0; Index < NumSubKs; Index++) {
    SubKNameLen = MaxSubKLen + 1;
    RegEnumKeyEx(hSubK, Index, SubKName, &SubKNameLen, NULL, NULL,
NULL, &LastWriteTime);
    DisplaySubKey(FullKName, SubKName, &LastWriteTime, Flags);
    if (Recursive) {
        _stprintf(FullSubKName, _T("%s\\%s"), FullKName, SubKName);
        TraverseRegistry(hSubK, FullSubKName, SubKName, Flags);
    }
}
_tprintf(_T("\n"));
free(SubKName);
free(ValName);
free(Val);
RegCloseKey(hSubK);
return TRUE;

```

```

}
BOOL DisplayPair(LPTSTR ValueName, DWORD ValueType, LPBYTE Value,
DWORD ValueLen, LPBOOL Flags)
/* Функция, отображающая пары "имя-значение". */
{
    LPBYTE pV = Value;
    DWORD i;
    _tprintf(_T("\nValue: %s = "), ValueName);
    switch (ValueType) {
        case REG_FULL_RESOURCE_DESCRIPTOR: /* 9: описание оборудования.
        */
        case REG_BINARY: /* 3: Любые двоичные данные. */
            for (i = 0; i < ValueLen; i++, pV++) _tprintf (_T (" %x"), *pV);
            break;
        case REG_DWORD: /* 4: 32-битовое число. */
            _tprintf(_T ("%x"), (DWORD)*Value);
            break;
        case REG_MULTI_SZ: /*7: массив строк, завершающихся нулевым
        символом.*/
        case REG_SZ: /* 1: строка, завершающаяся нулевым символом. */
            _tprintf(_T("%s"), (LPTSTR)Value);
            break;
        /* ... Несколько других типов ... */
    }
    return TRUE;
}

BOOL DisplaySubKey(LPTSTR KeyName, LPTSTR SubKeyName, PFILETIME
pLastWrite, LPBOOL Flags) {
    BOOL Long = Flags[1];
    SYSTEMTIME SysLastWrite;
    _tprintf(_T("\nSubkey: %s"), KeyName);
    if (_tcslen(SubKeyName) > 0) _tprintf (_T ("\\%s "), SubKeyName);
    if (Long) {
        FileTimeToSystemTime(pLastWrite, &SysLastWrite);
        _tprintf(_T("%02d/%02d/%04d %02d:%02d:%02d"), SysLastWrite.wMonth,
        SysLastWrite.wDay, SysLastWrite.wYear, SysLastWrite.wHour,
        SysLastWrite.wMinute, SysLastWrite.wSecond);
    }
    return TRUE;
}

```

**Задание 2.** Воспроизведите текст программы. Выполните его.

Сформируйте список разделов и содержимого реестра. Выведите список разделов и содержимое реестра.

**Задание 3.** Подготовить отчет о выполненной работе

### Критерии оценивания:

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

### Практические занятия № 6, №7

**Тема раздела:** Обработка исключений

**Тема практической работы 6:** Обработка ошибок как исключений

**Тема практической работы 7:** Использование обработчиков завершения для повышения качества программ

**Цель:** ознакомиться с обработкой исключений.

**Планируемые результаты:**

**уметь:** Обрабатывать ошибки как исключений

**знать:** понятие "исключение".

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Ознакомиться с теоретическими положениями

Книга: Системное программирование в среде Windows

#### **Ошибки и исключения**

Под ошибками понимаются исключительные ситуации, которые время от времени могут возникать в известных местах программы. Так, обнаружение ошибок, возникающих во время выполнения системных вызовов, и немедленный вывод сообщений о них должны предусматриваться логикой работы самой программы. Поэтому программисты, как правило, явно включают в программный код участки, ответственные, например, за тестирование успешности завершения операции чтения данных из файла. В



главе 2 для диагностики ошибок и принятия соответствующих мер была разработана функция ReportError.

С другой стороны, исключения могут возникать практически в любом месте программы, и поэтому организация явной проверки всех исключений невозможна или практически нецелесообразна. Примерами подобных ситуаций могут служить попытки деления на ноль или обращения к недоступным областям памяти.

Вместе с тем, указанные различия между ошибками и исключениями являются довольно условными. Windows позволяет управлять генерацией исключений, возникающих в случае нехватки памяти при ее распределении с использованием функций HeapAlloc и HeapCreate. Этот процесс описан в главе 5. Помимо этого, программы могут генерировать собственные исключения с кодами, определяемыми программистом, используя для этого функцию RaiseException, о чем далее будет говориться.

Обработчики исключений обеспечивают удобный механизм выхода из внутренних блоков или функций под управлением программы без использования операторов перехода goto или longjmp. Такая возможность оказывается особенно полезной, если блок получил доступ к таким, например, ресурсам, как открытые файлы, память или объекты синхронизации, поскольку обработчик может взять на себя задачу освобождения этих ресурсов. Возможно также продолжение работы программы после выполнения кода обработчика исключений, а не ее обязательное завершение. Кроме того, после выхода из блока программа может восстанавливать прежнее состояние системы, например маску FP-исключений. Именно в этом ключе обработчики используются во многих наших примерах.

### **Исключения, генерируемые приложением**

Существует возможность формирования исключений в любой точке программы в процессе ее выполнения с помощью функции RaiseException. Это позволяет программе обнаруживать и обрабатывать возникающие ошибки как исключения.

`VOID RaiseException(DWORD dwExceptionCode, DWORD dwExceptionFlags, DWORD cArguments, CONST DWORD *lpArguments)`

Параметры

dwExceptionCode — код исключения, определяемый пользователем. Бит 28 использовать нельзя, так как он зарезервирован системой. Для кода ошибки отводятся биты 27—0 (то есть все слово, кроме самого старшего шестнадцатеричного разряда). Бит 29 должен быть установлен, чтобы показать, что данное исключение имеет "пользовательскую" природу (а не относится к числу тех, которые предусмотрела Microsoft). В битах 31—30 содержится код серьезности ошибки, принимающий приведенные ниже значения, в которых результирующая старшая шестнадцатеричная цифра кода исключения представлена с установленным битом 29.

- 0 — успешное выполнение (старшая шестнадцатеричная цифра кода исключения равна 2).

- 1 — информационный код (старшая шестнадцатеричная цифра кода исключения равна 6).
- 2 — предупреждение (старшая шестнадцатеричная цифра кода исключения равна A).
- 3 — ошибка (старшая шестнадцатеричная цифра кода исключения равна E).

`dwExceptionFlags` — обычно устанавливается равным 0, тогда как установка значения `EXCEPTION_NONCONTINUABLE` будет указывать на то, что выражение фильтра не должно возвращать значение `EXCEPTION_CONTINUE_EXECUTION`; при попытке это сделать будет немедленно сгенерировано исключение `EXCEPTION_NONCONTINUABLE_EXCEPTION`.

`lpArguments` — этот указатель, если он не равен `NULL`, указывает на массив размера `cArguments` (третий параметр), содержащий 32-битовые значения, которые должны быть переданы выражению фильтра. Максимально возможное число этих значений ограничивается значением `EXCEPTION_MAXIMUM_PARAMETERS`, которое в настоящее время установлено равным 15. Для доступа к этой структуре следует использовать функцию `GetExceptionInformation`.

Заметьте, что невозможно сгенерировать исключение в другом процессе. В то же время, при весьма ограниченных условиях для этой цели могут быть использованы обработчики управляющих сигналов консоли

### **Пример: обработка ошибок как исключений**

В предыдущих примерах для обработки ошибок при выполнении системных вызовов и других ошибок используется функция `ReportError`. Эта функция прекращает выполнение процесса, если программист указал, что данная ошибка является критической. Вместе с тем, такой подход препятствует нормальному выходу из программы и не обеспечивает возможность продолжения работы программы после устранения последствий ошибки. Так, после отказа от задачи, которая привела к возникновению сбоя, может потребоваться уничтожение временных файлов, созданных в процессе работы программы, или переход программы к выполнению других задач. Функции `ReportError` присущи и другие ограничения, перечень которых приводится ниже.

- Даже в тех случаях, когда было бы достаточно прекратить выполнения только одного потока, критическая ошибка приводит к остановке всего процесса .
- Вместо завершения процесса может оказаться желательным продолжение выполнения программы.
- Во многих случаях становится невозможным освобождение ресурсов синхронизации, например мьютексов.

При прекращении выполнения процесса (но не потоки) открытые дескрипторы будут закрываться, однако при этом необходимо учитывать другие отрицательные факторы.

Решение заключается в написании новой функции — `ReportException`. Если ошибка не является критической, эта функция вызывает функцию `ReportError`, которая выводит сообщение об ошибке. В случае же возникновения критической ошибки будет сгенерировано исключение. Система будет использовать обработчик исключений из вызывающего `try`-блока, и поэтому в действительности характер исключения может быть не критическим, если обработчик предоставляет программе возможность восстановиться после сбоя. По существу, функция `ReportException` дополняет обычные программные методы защиты от ошибок, ранее ограниченные функцией `ReportError`. В случае обнаружения ошибки обработчик позволяет программе продолжить свою работу после выполнения необходимых восстановительных действий. Эти возможности иллюстрирует программа 5.2.

Функция `ReportException` представлена в программе 5.1. Необходимые определения и заголовочные файлы не указаны, поскольку эта функция находится в том же модуле исходного кода, что и функция `ReportError`.

Программа 5.1. `ReportException`: функция вывода сообщений об исключениях

```
/* Расширение функции ReportError для генерации формируемого
приложением кода исключения вместо прекращения выполнения процесса. */
VOID ReportException(LPCTSTR UserMessage, DWORD ExceptionCode)
/* Вывести сообщение о не критической ошибке. */
{
    ReportError(UserMessage, 0, TRUE);
    /* Если ошибка критическая, сгенерировать исключение. */
    if (ExceptionCode != 0) RaiseException((0x0FFFFFFF & ExceptionCode) |
0xE0000000, 0, 0, NULL);
    return;
}
```

Функция `ReportException` используется в нескольких последующих примерах.

Модель сигналов, используемая в UNIX, значительно отличается от SEH. Сигналы могут быть пропущены или игнорированы, и логика их работы иная. Тем не менее, у этих моделей имеются и общие черты.

Значительная часть поддержки обработки сигналов в UNIX обеспечивается библиотекой C, ограниченная версия которой доступна также под управлением Windows. Во многих случаях в программах Windows вместо сигналов можно воспользоваться обработчиками управляющих сигналов консоли, описанными в конце данной главы.

Некоторые сигналы соответствуют исключениям Windows.

Перечень в некоторой мере ограниченных соответствий "сигнал-исключение" представлен ниже:

- `SIGILL` — `EXCEPTION_PRIV_INSTRUCTION`
- `SIGSEGV` — `EXCEPTION_ACCESS_VIOLATION`

• SIGFPE — семь различных исключений, связанных с выполнением операций над числами с плавающей точкой, например EXCEPTION\_FLT\_DIVIDE\_BY\_ZERO

• SIGUSR1 и SIGUSR2 — исключения, определяемые приложением. Функции RaiseException соответствует функция библиотеки C raise.

В Windows сигналы SIGILL, SIGSEGV и SIGFPE не генерируются, хотя функция raise может генерировать один из них. Сигнал SIGINT в Windows не поддерживается.

Функция UNIX kill (kill не входит в состав стандартной библиотеки C), которая посылает сигнал другому процессу, может быть сопоставлена функции Windows GenerateConsoleCtrlEvent. Для ограниченного варианта SIGKILL в Windows имеются аналоги в виде функций TerminateProcess и TerminateThread, с помощью которых один процесс (или поток) может уничтожить другой, хотя при использовании этих функций необходимо соблюдать осторожность.

### **Обработчики завершения**

Обработчики завершения служат в основном тем же целям, что и обработчики исключений, но выполняются, когда поток покидает блок в результате нормального выполнения программы, а также когда возникает исключение. С другой стороны, обработчик завершения не может распознавать исключения.

Обработчик завершения строится с использованием ключевого слова `_finally` в операторе `try...finally`. Структура этого оператора аналогична структуре оператора `try...finally`, но в ней отсутствует выражение фильтра. Как и обработчики исключений, обработчики завершения предоставляют удобные возможности для закрытия дескрипторов, освобождения ресурсов, восстановления масок и выполнения иных действий, направленных на восстановление известного состояния системы после выхода из блока. Например, программа может выполнять операторы `return` внутри блока, оставляя всю работу по "уборке мусора" обработчику завершения. Благодаря этому отпадает необходимость во включении кода очистки в код самого блока или переходе к коду очистки при помощи оператора `goto`.

```
__try {  
    /* Блок кода. */  
} _finally {  
    /* Обработчик завершения (блок finally). */  
}
```

### **Выход из try-блока**

Обработчик завершения выполняется всякий раз, когда в соответствии с логикой программы осуществляется выход из `try`-блока по одной из следующих причин:

- Достижение конца `try`-блока и "проваливание" в обработчик завершения.
- Выполнение одного из следующих операторов таким образом, что происходит выход за пределы блока:

`return`

break  
goto[\[19\]](#)  
longjmp  
continue  
\_\_leave[\[20\]](#)

- Исключение.

### Аварийное завершение

Любое завершение выполнения программы по причинам, отличным от достижения конца try-блока и "проваливания вниз" или выполнения оператора \_\_leave, считается аварийным завершением. Результатом выполнения оператора \_\_leave является переход в конец блока \_\_try и передача управления вниз по тексту программы, что намного эффективнее простого использования оператора goto, поскольку не требует разворачивания стека. Для определения того, каким образом завершилось выполнение try-блока, в обработчике завершения используется следующая функция:

BOOL AbnormalTermination(VOID)

При аварийном завершении выполнения блока эта функция возвращает значение TRUE, при нормальном — FALSE.

#### Примечание

Завершение будет считаться аварийным, даже если, например, последним оператором try-блока был оператор return.

Выполнение обработчика завершения и выход из него

Обработчик завершения, или блок \_\_finally, выполняется в контексте блока или функции, работу которых он отслеживает. Управление может переходить от оператора завершения к следующему оператору. Существует и другая возможность, когда обработчик завершения выполняет оператор передачи управления (return, break, continue, goto, longjmp или \_\_leave). Еще одной возможностью является выход из обработчика по причине возникновения исключения.

Сочетание блоков finally и except

Один try-блок может иметь только один блок finally или только один блок except, но не может иметь оба указанных блока одновременно. Поэтому нижеприведенный код вызовет появление ошибок на стадии компиляции.

```
__try {  
    /* Блок контролируемого кода. */  
} __except (filter_expression) {  
    /* Блок обработчика исключений. */  
} __finally {  
    /* Так делать нельзя! Это приведет к ошибке на стадии компиляции. */  
}
```

Вместе с тем, допускается вложение одного блока в другой, что используется довольно часто. Нижеприведенный код является вполне работоспособным и обеспечивает гарантированное удаление временных файлов при выходе из цикла под управлением программы или в результате

возникновения исключения. Эта методика оказывается удобной и в тех случаях, когда требуется обеспечить гарантированную отмену блокирования файлов, что будет использовано в программе 5.2. Кроме того, в коде имеется внутренний блок `try...except`, размещенный в том месте программы, где выполняются вычисления, в которых участвуют вещественные числа.

```
__try { /* Внешний блок try-except. */
while (...) __try { /* Внутренний блок try-finally. */
    hFile = CreateFile(TempFile, ...);
    if(...) __try { /* Внутренний блок try-except. */
        /* Разрешить FP-исключения. Выполнить вычисления. */
        ...
    } __except(EXCEPTION_EXECUTE_HANDLER) {
        ... /* Обработать FP-исключение. */
        _clearfp();
    }
    ... /* Обработка исключений, не являющихся FP-исключениями. */
} __finally { /* Конец цикла while. */
    /* Выполняется на КАЖДОЙ итерации цикла. */
    CloseHandle(hFile);
    DeleteFile(TempFile);
}
} __except (filter-expression) {
/* Обработчик исключений. */
}
```

### **Пример: использование обработчиков завершения для повышения качества программ**

Обработчики исключений и завершения позволяют повысить надежность программ как за счет упрощения процедуры восстановления программы после возникновения ошибок и исключений, так и за счет гарантированного освобождения ресурсов и отмены блокирования файлов в критических ситуациях.

В программе `tourper` (программа 5.2) эти моменты иллюстрируются с привлечением идей, почерпнутых в программном коде предшествующих примеров. `tourper` обрабатывает несколько файлов, имена которых указываются в командной строке, переписывая их с преобразованием всех букв в верхний регистр. Имена преобразованных файлов получаются путем добавления префикса `UC_` к исходным именам, и согласно "спецификации" программы запись поверх существующих файлов не производится. Преобразование файлов осуществляется в памяти машины, поэтому для каждого файла выделяется большая буферная область (достаточная для размещения всего файла). Кроме того, чтобы исключить любую возможность изменения файлов другими процессами, а также для того, чтобы вновь создаваемые выходные файлы строго соответствовали преобразованным входным файлам, оба вида файлов блокируются во время обработки. Понятно, что на каждой стадии обработки существует вероятность

возникновения самых различных сбойных ситуаций, но в программе должна быть предусмотрена защита от подобных ошибок, и она должна располагать средствами, позволяющими ей восстановить свое нормальное состояние и попытаться обработать все остальные файлы, имена которых были указаны в командной строке. Программа 5.2 решает все эти задачи, обеспечивая разблокирование файлов во всех необходимых случаях без применения громоздкой логики операторов ветвления, к которым пришлось бы прибегнуть, если бы не были использованы средства SEH. Более подробные комментарии к программе содержатся в программном коде, находящемся на Web-сайте книги.

Программа 5.2. toupper: обработка файлов с восстановлением нормального состояния программы после сбоев

```
/* Глава 4. Команда toupper. */
```

```
/* Преобразование содержимое одного и более файлов с заменой всех букв на прописные. Имя выходного файла получается из имени входного файла добавлением к нему префикса UC_. */  
#include "EvryThng.h"
```

---

```
int _tmain(DWORD argc, LPTSTR argv[]) {  
    HANDLE hIn = INVALID_HANDLE_VALUE, hOut =  
    INVALID_HANDLE_VALUE;  
    DWORD FileSize, nXfer, iFile, j;  
    CHAR OutFileName [256] = "", *pBuffer = NULL;  
    OVERLAPPED ov = {0, 0, 0, 0, NULL}; /* Используется для блокирования  
    файлов. */  
    if (argc <= 1) ReportError(_T("Использование: toupper файлы"), 1, FALSE);  
    /* Обработать все файлы, указанные в командной строке. */  
    for (iFile = 1; iFile < argc; iFile++) __try { /* Блок исключений. */  
        /* Все дескрипторы файлов недействительны, pBuffer == NULL, а файл  
        OutFileName пуст. Выполнение этих условий обеспечивается обработчиками.  
        */  
        _stprintf(OutFileName, "UC_%s", argv[iFile]);  
        __try { /* Внутренний блок try-finally. */  
            /* Ошибка на любом шаге сгенерирует исключение, и следующий */  
            /* файл будет обрабатываться только после "уборки мусора". */  
            /* Объем работы по очистке зависит от того, в каком месте */  
            /* программы возникла ошибка. */  
            /* Создать выходной файл (завершается с ошибкой, если файл уже  
            существует). */  
            hIn = CreateFile(argv[iFile], GENERIC_READ, 0, NULL, OPEN_EXISTING,  
            0, NULL);  
            if (hIn == INVALID_HANDLE_VALUE) ReportException(argv[iFile], 1);  
            FileSize = GetFileSize(hIn, NULL);  
            hOut = CreateFile(OutFileName, GENERIC_READ | GENERIC_WRITE, 0,  
            NULL, CREATE_NEW, 0, NULL);
```

```

if (hOut == INVALID_HANDLE_VALUE) ReportException(OutFileName, 1);
/* Распределить память под содержимое файла. */
pBuffer = malloc(FileSize);
if (pBuffer == NULL) ReportException(_T("Ошибка при распределении
памяти"), 1);
/* Блокировать оба файла для обеспечения целостности копии. */
if (!LockFileEx(hIn, LOCKFILE_FAIL_IMMEDIATELY, 0, FileSize, 0, &ov)
ReportException(_T("Ошибка при блокировании входного файла"), 1);
if (!LockFileEx(hOut, LOCKFILE_EXCLUSIVE_LOCK |
LOCKFILE_FAIL_IMMEDIATELY, 0, FileSize, 0, &ov)
ReportException(_T("Ошибка при блокировании выходного файла "), 1);
/* Считать данные, преобразовать их и записать в выходной файл. */
/* Освободить ресурсы при завершении обработки или возникновении */
/* ошибки; обработать следующий файл. */
if (!ReadFile(hIn, pBuffer, FileSize, &nXfer, NULL))
ReportException(_T("Ошибка при чтении файла"), 1);
for (j = 0; j < FileSize; j++) /* Преобразовать данные. */
if (isalpha(pBuffer [j])) pBuffer[j] = toupper(pBuffer [j]);
if(WriteFile(hOut, pBuffer, FileSize, &nXfer, NULL))
ReportException(_T("Ошибка при записи в файл"), 1);
} __finally {
/*Освобождение блокировок, закрытие дескрипторов файлов,*/
/*освобождение памяти и повторная инициализация */
/*дескрипторов и указателя. */
if (pBuffer != NULL) free (pBuffer);
pBuffer = NULL;
if (hIn != INVALID_HANDLE_VALUE) {
UnlockFileEx(hIn, 0, FileSize, 0, &ov);
CloseHandle(hIn);
hIn = INVALID_HANDLE_VALUE;
}
if (hOut != INVALID_HANDLE_VALUE) {
UnlockFileEx(hOut, 0, FileSize, 0, &ov);
CloseHandle(hOut);
hOut = INVALID_HANDLE_VALUE;
}
_tcscpy(OutFileName, _T(""));
}
}
/* Конец основного цикла обработки файлов и блока try. */
/* Обработчик исключений для тела цикла. */
__except(EXCEPTION_EXECUTE_HANDLER) {
_tprintf(_T("Ошибка при обработке файла %s\n"), argv[iFile]);
DeleteFile (OutFileName);
}

```



```
_tprintf(_T("Обработаны все файлы, кроме указанных выше \n"));
return 0;
}
```

**Задание 2.** Воспроизведите текст программы. Выполните его.

**Задание 3.** Подготовить отчет о выполненной работе

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## **Практическое занятие № 8**

**Тема раздела:** Обработка исключений

**Тема практической работы:** Обработчик управляющих сигналов консоли

**Цель:** ознакомиться с обработкой исключений.

**Планируемые результаты:**

**уметь:** Обрабатывать управляющие сигналы консоли

**знать:** понятие управляющие сигналы консоли.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Ознакомиться с теоретическими положениями

Книга: Системное программирование в среде Windows

### **Обработчики управляющих сигналов консоли**

Обработчики исключений могут реагировать на самые разнообразные события, но они не в состоянии обнаруживать такие ситуации, как выход пользователя из системы или нажатие комбинации клавиш <Ctrl+C> на

клавиатуре с целью прекращения выполнения программы. Для обработки таких событий требуются обработчики управляющих сигналов консоли.

Функция `SetConsoleCtrlHandler` позволяет одной или нескольким указанным функциям выполняться в ответ на получение сигналов `Ctrl-c`, `Ctrl-break` или одного из трех других сигналов, связанных с консолью. Функция `GenerateConsoleCtrlEvent`, описанная в главе 6, также генерирует эти сигналы, а, кроме того, все эти сигналы могут посылаются другим процессам, совместно использующим ту же консоль. Обработчиками сигналов являются указанные пользователем функции, которые возвращают булевские значения и принимают единственный аргумент типа `DWORD`, идентифицирующий фактический сигнал.

С одним сигналом могут быть ассоциированы несколько обработчиков, причем обработчики можно добавлять и удалять. Функция, которая используется для добавления и удаления обработчиков, имеет следующий вид:

```
BOOL SetConsoleCtrlHandler(PHANDLER_ROUTINE HandlerRoutine,  
BOOL Add)
```

Значению флага `Add`, равному `TRUE`, соответствует добавление процедуры обработчика, в противном случае происходит удаление процедуры из списка процедур обработки управляющих сигналов консоли. Заметьте, что тип сигнала при вызове функции не конкретизируется. Тестирование с целью проверки того, какой именно сигнал получен, должен выполнять сам обработчик.

Процедура обработчика возвращает булевское значение и принимает единственный параметр типа `DWORD`, идентифицирующий фактический сигнал. Использованное в объявлении имя обработчика (`HandlerRoutine`) является заменителем, и программист может выбирать его по своему усмотрению.

Ниже приводятся дополнительные полезные сведения, касающиеся использования обработчиков управляющих сигналов консоли.

- Если значение параметра `HandlerRoutine` равно `NULL`, а параметра `Add` — `TRUE`, то сигналы `Ctrl-c` будут игнорироваться.
- Если при вызове функции `SetConsoleMode` был задан параметр `ENABLE_PROCESSED_INPUT` (глава 2), то комбинация `<Ctrl+C>` будет обрабатываться не как сигнал, а как клавиатурный ввод.
- Процедура обработчика фактически выполняется как независимый поток (см. главу 7) внутри процесса. При этом выполнение основной программы, как показано в следующем примере, не приостанавливается.
- Формирование исключения в обработчике не вызовет исключения в потоки, выполнение которого было прервано, поскольку исключения применяются только к потокам, а не к процессу в целом. Если вы хотите организовать связь с прерванным потоком, используйте переменную, как в следующем примере, или метод синхронизации.

Между исключениями и сигналами существует важное отличие. Сигналы применяются к процессу в целом, тогда как исключения — только к потоку, выполняющему код, в котором возникло исключение.

**BOOL HandlerRoutine(DWORD dwCtrlType)**

**dwCtrlType** идентифицирует фактический сигнал (или событие) и может принимать одно из следующих пяти значений:

1. **CTRL\_C\_EVENT** указывает на то, что комбинация <Ctrl+C> должна восприниматься как клавиатурный ввод.
2. **CTRL\_CLOSE\_EVENT** указывает на закрытие окна консоли.
3. **CTRL\_BREAK\_EVENT** указывает на сигнал Ctrl-break.
4. **CTRL\_LOGOFF\_EVENT** указывает на выход пользователя из системы.
5. **CTRL\_SHUTDOWN\_EVENT** указывает на завершение работы системы.

Обработчик сигналов может выполнять операции по "уборке мусора" точно так же, как это делают обработчики исключений и завершения. В случае успешной обработки сигнала обработчик должен вернуть значение **TRUE**. Если обработчик возвращает значение **FALSE**, выполняется следующая функция обработчика из числа тех, что указаны в списке. Обработчики сигналов выполняются в порядке, обратном порядку их установки, так что первым будет выполняться самый последний из установленных обработчиков, а системный обработчик будет выполняться самым последним.

### **Пример: обработчик управляющих сигналов консоли**

В программе 7 организован бесконечный цикл, в котором каждые 5 секунд вызывается функция **Веер**, подающая звуковой сигнал. Пользователь может завершить выполнение программы, нажав комбинацию клавиш <Ctrl+C> или закрыв консоль. Процедура обработчика выводит на экран сообщение, выжидает 10 секунд, после чего, казалось бы, выполнение программы должно завершиться с возвратом значения **TRUE**. Однако в действительности основная программа обнаруживает флаг **Exit** и останавливает процесс. Это демонстрирует параллельную природу выполнения процедуры обработчика; заметьте, что объем выходной информации обработчика сигналов зависит от временных характеристик сигнала. Обработчики управляющих сигналов консоли будут использоваться также в примерах, приводимых в следующих главах.

Обратите внимание на использование макроса **WINAPI**; он применяется к пользовательским функциям, передаваемым в качестве аргументов функциям **Windows**, чтобы гарантировать выполнение соответствующих соглашений о вызовах. Этот макрос определен в заголовочном файле **Microsoft C WTYPES.H**.

Программа 7. **Ctrlc**: программа обработки сигналов

```
/* Глава 4. Ctrlc.c */
```

```
/* Перехватчик событий консоли. */
```

```
#include "EvryThng.h"
```

```

static BOOL WINAPI Handler(DWORD CtrlEvent); /* См. WTYPES.H. */
volatile static BOOL Exit = FALSE;
int _tmain(int argc, LPTSTR argv[])
/* Периодическая подача звукового сигнала до поступления сигнала о
прекращении выполнения. */
{
/* Добавить обработчик событий. */
if (!SetConsoleCtrlHandler(Handler, TRUE)) ReportError(_T("Ошибка при
установке обработчика событий."), 1, TRUE);
while (!Exit) {
Sleep(5000); /* Подача звукового сигнала каждые 5 секунд. */
Beep(1000 /* Частота. */, 250 /* Длительность. */);
}
_tprintf(_T("Прекращение выполнения программы по требованию.\n"));
return 0;
}
BOOL WINAPI Handler (DWORD CtrlEvent) {
Exit = TRUE;
switch (CntrlEvent) {
/* Увидите ли вы второе сообщения обработчика, зависит от соотношения
временных параметров. */
case CTRL_C_EVENT:
_tprintf(_T("Получен сигнал Ctrl-c. Выход из обработчика через 10
секунд.\n"));
Sleep(4000); /* Уменьшите это значение, чтобы получить другой эффект. */
_tprintf(_T("Выход из обработчика через 6 секунд.\n"));
Sleep(6000); /* Попробуйте уменьшить и это значение. */
return TRUE; /* TRUE указывает на успешную обработку сигнала. */
case CTRL_CLOSE_EVENT:
_tprintf(_T("Выход из обработчика через 10 секунд.\n"));
Sleep(4000);
_tprintf(_T("Выход из обработчика через 6 секунд.\n"));
Sleep(6000); /* Попробуйте уменьшить и это значение. */
return TRUE; /* Попробуйте вернуть FALSE. Приводит ли это
к изменению поведения программы? */
default:
_tprintf(_T("Событие: %d. Выход из обработчика через 10 секунд.\n"),
CntrlEvent);
Sleep(4000);
_tprintf(_T("Выход из обработчика через 6 секунд.\n"));
Sleep(6000);
return TRUE;
}
}

```

**Задание 2.** Воспроизведите текст программы. Выполните его.

### Задание 3. Подготовить отчет о выполненной работе

#### Критерии оценивания:

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

### Практическое занятие № 9

**Тема раздела:** Процессы

**Тема практической работы:** Управление процессами

**Цель:** ознакомиться с управлением процессами.

**Планируемые результаты:**

**уметь:** управлять процессами.

**знать:** понятие "процесс".

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Ознакомиться с теоретическими положениями

Книга: Системное программирование в среде Windows

#### Процессы и потоки Windows

Внутри каждого процесса могут выполняться одна или несколько потоков, и именно поток является базовой единицей выполнения в Windows. Выполнение потоков планируется системой на основе обычных факторов: наличие таких ресурсов, как CPU и физическая память, приоритеты, равнодоступность ресурсов и так далее. Начиная с версии NT4, в Windows поддерживается симметричная многопроцессорная обработка (Symmetric Multiprocessing, SMP), позволяющая распределять выполнение потоков между отдельными процессорами, установленными в системе.

С точки зрения программиста каждому процессу принадлежат ресурсы, представленные следующими компонентами:

- Одна или несколько потоков.
- Виртуальное адресное пространство, отличное от адресных пространств других процессов, если не считать областей памяти, распределенных явным образом для совместного использования (разделения) несколькими процессами. Заметьте, что разделяемые отображенные файлы совместно используют физическую память, тогда как разделяющие их процессы используют различные виртуальные адресные пространства.
- Один или несколько сегментов кода, включая код DLL.
- Один или несколько сегментов данных, содержащих глобальные переменные.
- Строки, содержащие информацию об окружении, например, информацию о текущем пути доступа к файлам.
- Куча процесса.
- Различного рода ресурсы, например, дескрипторы открытых файлов и другие кучи.

Поток разделяет вместе с процессом код, глобальные переменные, строки окружения и другие ресурсы. Каждый поток планируется независимо от других и располагает следующими элементами:

- Стек, используемый для вызова процедур, прерываний и обработчиков исключений, а также хранения автоматических переменных.
- Локальные области хранения потока (Thread Local Storage, SLT) — массивы указателей, используя которые каждый поток может создавать собственную уникальную информационную среду.
- Аргумент в стеке, получаемый от создающего потока, который обычно является уникальным для каждого потока.
- Структура контекста, поддерживаемая ядром системы и содержащая значения машинных регистров.

На рис. показан процесс с несколькими потоками. Рисунок является схематическим, поэтому на нем не указаны фактические адреса памяти и не соблюдены масштабы.

В данной главе показано, как работать с процессами, состоящими из единственного потока.

### **Примечание**

*Рисунок Процесс и его потоки* является высокоуровневым с точки зрения программиста представлением процесса. В действительности эта картина должна быть дополнена множеством технических деталей и особенностями реализации. Более подробную информацию заинтересованные читатели могут найти в книге Соломона (Solomon) и Руссиновича (Russeinovich) *Inside Windows 2000*.

Процессы UNIX сопоставимы с процессами Windows, имеющими единственный поток.

Реализации UNIX недавно пополнились потоками POSIX Pthreads, которые в настоящее время используются почти повсеместно.

Наверное, можно было бы даже не напоминать о том, что понятие потоков не является новым, и их различные реализации предлагаются поставщиками уже на протяжении целого ряда лет. Однако потоки Pthreads являются самым распространенным стандартом, в то время как коммерческие реализации потоков являются устаревшими.

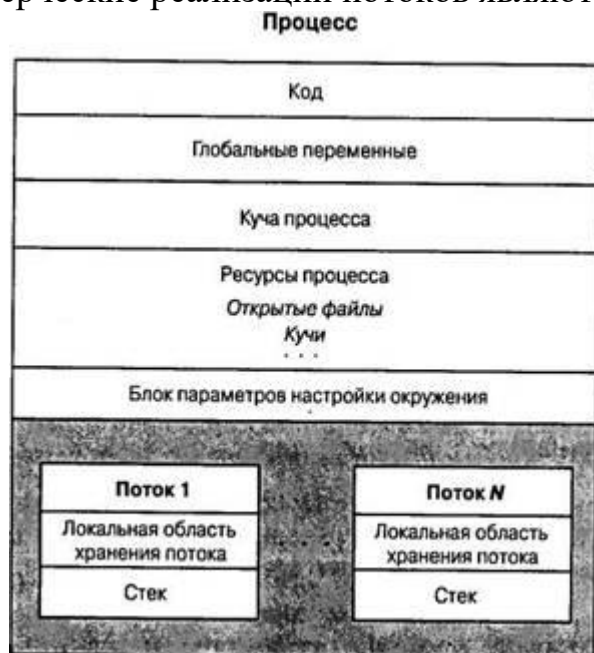


Рис. Процесс и его потоки

Одной из важнейших функций Windows, обеспечивающих управление процессами, является функция `CreateProcess`, которая создает новый процесс с единственным потоком. При вызове этой функции требуется указать имя файла исполняемой программы.

Обычно принято говорить о процессах-предках, или родительских процессах (parent processes), и процессах-потомках, или дочерних процессах (child processes), однако между процессами Windows эти отношения фактически не поддерживаются. Использование данной терминология является просто удобным способом выражения того факта, что один процесс порождается другим.

Гибкие и мощные возможности функции `CreateProcess` обеспечиваются ее десятью параметрами. На первых порах для упрощения работы целесообразно использовать значения параметров, заданные по умолчанию. Точно так же, как и в случае функции `CreateFile`, имеет смысл подробно рассмотреть каждый из параметров функции `CreateProcess`. Благодаря этому изучить другие аналогичные функции вам будет гораздо легче.

Прежде всего, заметьте, что возвращаемое значение функции не является дескриптором типа `HANDLE`; вместо этого функция возвращает два отдельных дескриптора, по одному для процесса и потока, передавая их в структуре, которая указывается при вызове функции. Эти дескрипторы относятся к создаваемому функцией `CreateProcess` новому процессу и его основному (primary) потоку. Во избежание утечки ресурсов в процессе работы с примерами программ тщательно следите за своевременным закрытием обоих дескрипторов, когда они вам больше не нужны;

забывчивость в отношении закрытия дескрипторов потоков является одной из самых распространенных ошибок. Закрытие дескриптора потока не приводит к прекращению ее выполнения; функция `CloseHandle` лишь удаляет ссылку на поток внутри процесса, вызвавшего функцию `CreateProcess`.

`BOOL CreateProcess(lpApplicationName, LPTSTR lpCommandLine, LPSECURITY_ATTRIBUTES lpsaProcess, LPSECURITY_ATTRIBUTES lpsaThread, BOOL bInheritHandles, DWORD dwCreationFlags, LPVOID lpEnvironment, LPCTSTR lpCurDir, LPSTARTUPINFO lpStartupInfo, LPPROCESS_INFORMATION lpProcInfo)`

Возвращаемое значение: в случае успешного создания процесса и потока — `TRUE`, иначе — `FALSE`.

### Параметры

Некоторые параметры потребуют дальнейшего подробного обсуждения в следующих разделах, тогда как смысл многих других станет для вас более понятным при рассмотрении примеров программ.

`lpApplicationName` и `lpCommandLine` (последний указатель имеет тип `LPTSTR`, а не `LPCTSTR`) — используются вместе для указания исполняемой программы и аргументов командной строки, о чем говорится в следующем разделе.

`lpsaProcess` и `lpsaThread` — указатели на структуры атрибутов защиты процесса и потока. Значениям `NULL` соответствует использование атрибутов защиты, заданных по умолчанию, и именно эти значения будут использоваться нами вплоть до главы 15, посвященной рассмотрению средств безопасности Windows.

`bInheritHandles` — показывает, наследует ли новый процесс наследуемые открытые дескрипторы (файлов, отображений файлов и так далее) из вызывающего процесса. Наследуемые дескрипторы имеют те же атрибуты, что и исходные, и их обсуждение будет продолжено в одном из следующих разделов.

`dwCreationFlags` — может объединять в себе несколько флаговых значений, включая следующие:

- `CREATE_SUSPENDED` — указывает на то, что основной поток будет создан в приостановленном состоянии и начнет выполняться лишь после вызова функции `ResumeThread`.

- `DETACHED_PROCESS` и `CREATE_NEW_CONSOLE` — взаимоисключающие значения, которые не должны устанавливаться оба одновременно. Первый флаг означает создание нового процесса, у которого консоль отсутствует, а второй — процесса, у которого имеется собственная консоль. Если ни один из этих флагов не указан, то новый процесс наследует консоль родительского процесса.

- `Create_New_Process_Group` — указывает на то, что создаваемый процесс является корневым для новой группы процессов. Если все процессы, принадлежащие данной группе, разделяют общую консоль, то все они будут получать управляющие сигналы консоли (`Ctrl-C` или `Ctrl-break`).



Некоторые из флагов управляют приоритетами потоков нового процесса. Пока же нам будет достаточно использовать приоритет родительского процесса (этот режим устанавливается по умолчанию) или указывать значение `NORMAL_PRIORITY_CLASS`.

`lpEnvironment` — указывает на блок параметров настройки окружения нового процесса. Если задано значение `NULL`, то новый процесс будет использовать значения параметров окружения родительского процесса. Блок параметров содержит строки, в которых заданы пары "имя-значение", определяющие, например, пути доступа к файлам.

`lpCurDir` — указатель на строку, содержащую путь к текущему каталогу нового процесса. Если задано значение `NULL`, то в качестве текущего каталога будет использоваться рабочий каталог родительского процесса.

`lpStartupInfo` — указатель на структуру, которая описывает внешний вид основного окна и содержит дескрипторы стандартных устройств нового процесса. Используйте соответствующую информацию из родительского процесса, которую можно получить при помощи функции `GetStartupInfo`. Можно поступить и по-другому, обнулив структуру `STARTUPINFO` перед вызовом функции `CreateProcess`. Для указания стандартных устройств ввода, вывода информации и вывода сообщений об ошибках следует определить значения полей дескрипторов стандартных устройств (`hStdInput`, `hStdOutput` и `hStdError`) в структуре `STARTUPINFO`. Чтобы эти значения не игнорировались, следует задать для другого элемента этой же структуры, а именно, элемента `dwFlags`, значение `STARTF_USESTDHANDLES` и определить все дескрипторы, которые потребуются дочернему процессу. Убедитесь в том, что эти дескрипторы являются наследуемыми и что при вызове функции `CreateProcess` значение параметра `bInheritHandles` установлено равным `TRUE`. Более подробная информация по этому вопросу, сопровождаемая соответствующим примером, приводится в разделе "Наследуемые дескрипторы".

`lpProInfo` — указатель на структуру, в которую будут помещены возвращаемые функцией значения дескрипторов и глобальных идентификаторов процесса и потока. Структура `PROCESS_INFORMATION`, о которой идет речь, имеет следующий вид:

```
typedef struct PROCESS_INFORMATION {  
    HANDLE hProcess;  
    HANDLE hThread;  
    DWORD dwProcessId;  
    DWORD dwThreadId;  
} PROCESS_INFORMATION;
```

Зачем процессам и потокам нужны еще и дескрипторы, если они снабжаются глобальными идентификаторами (ID)? Глобальные идентификаторы остаются уникальными для данного объекта на протяжении всего времени его существования и во всех процессах, тогда дескрипторы процесса может быть несколько и каждый из которых может

характеризоваться собственным набором атрибутов, например определенными разрешениями доступа. В силу указанных причин одним функциям управления процессами требуется предоставлять идентификаторы процессов, а другим — дескрипторы. Кроме того, необходимость в дескрипторах процессов возникает при использовании универсальных функций, которые требуют указания дескрипторов. В качестве примера можно привести функции ожидания, обсуждаемые далее в этой главе, которые обеспечивают отслеживание переходов объектов различного типа, в том числе и процессов, указываемых с помощью дескрипторов, в определенные состояния. Точно так же, как и дескрипторы файлов, дескрипторы процессов и потоков должны закрываться сразу же после того, как необходимость в них отпала.

### **Примечание**

Новый процесс получает информацию об окружении, рабочем каталоге и иную информацию в результате вызова функции `CreateProcess`. По завершении этого вызова любые изменения характеристик родительского процесса никак не отразятся на дочернем процессе. Так, после вызова функции `CreateProcess` рабочий каталог родительского процесса может измениться, но на дочерний процесс это не окажет никакого влияния, если только он сам не сменит рабочий каталог. Оба процесса полностью независимы друг от друга.

Модели процесса в UNIX и Windows значительно отличаются друг от друга. Прежде всего, в Windows отсутствует эквивалент UNIX-функции `fork`, создающей копию родительского процесса, включая его пространство данных, кучу и стек. В Windows трудно добиться точной эмуляции `fork`, но как ни расценивать последствия этого ограничения, остается фактом, что проблемы с использованием функции `fork` существуют и в многопоточных системах UNIX, поскольку любые попытки создания точной реплики многопоточной системы с копиями всех потоков и объектов синхронизации, особенно в случае SMP-систем, приводят к возникновению множества трудностей. Поэтому в действительности функция `fork` вообще плохо подходит для многопоточных систем.

В то же время, функция `CreateProcess` аналогична обычной для UNIX цепочке последовательных вызовов функций `fork` и `exec1` (или одной из пяти остальных функций `exec`). В отличие от Windows пути доступа в UNIX определяются исключительно переменной среды `PATH`.

Как ранее уже отмечалось, отношения "предок-потомок" между процессами в Windows не поддерживаются. Так, выполнение дочернего процесса будет продолжаться даже после того, как завершится родительский процесс. Кроме того, в Windows отсутствуют группы процессов. Существует, однако, ограниченная форма группы процессов, в которой все процессы получают управляющие события консоли.

Процессы Windows идентифицируются как дескрипторами, так и идентификаторами процессов, тогда как в UNIX дескрипторы процессов отсутствуют.

### **Пример: параллельный поиск указанного текстового шаблона**

Настало время посмотреть на процессы Windows в действии. Приведенная ниже в качестве примера программа `grepMP` создает процессы для поиска указанного текстового шаблона в файлах, по одному процессу на каждый файл. Эта программа моделирует UNIX-утилиту `grep`, хотя используемая нами методика применима к любой программе, которая полагается на стандартный вывод. Рассматривайте программу поиска как "черный ящик" и считайте, что она является просто исполняемой программой, выполнение которой должно контролироваться родительским процессом.

Командная строка программы имеет следующий вид:

`grepMP шаблон F1 F2 ... FN`

Программа 6.1 выполняет следующие виды обработки:

- Для поиска указанного шаблона в каждом из входных файлов, от F1 до FN, используется отдельный процесс, запускающий один и тот же исполняемый модуль. Для каждого процесса программа создает командную строку такого вида: `grep шаблон FK`.
- Полю `hStdOut` структуры `STARTUPINFO` нового процесса присваивается значение дескриптора временного файла, который определяется как наследуемый.
- Программа организует ожидание завершения всех процессов поиска, используя для этого функцию `WaitForMultipleObjects`.
- По завершении всех процессов поиска осуществляется поочередный вывод результатов (временных файлов). Вывод временного файла осуществляет процесс, выполняющий утилиту `cat` (программа 2.3).
- Возможности функции `WaitForMultipleObjects` ограничиваются лишь максимально допустимым количеством дескрипторов, которое устанавливается значением `MAXIMUM_WAIT_OBJECTS` (64), поэтому она вызывается многократно.
- Для определения успешности попытки нахождения данным процессом заданного шаблона программа использует код завершения процесса `grep`.

Порядок обработки файлов программой 8 иллюстрируется на рис. *Поиск текстового шаблона в файлах с использованием нескольких процессов.*

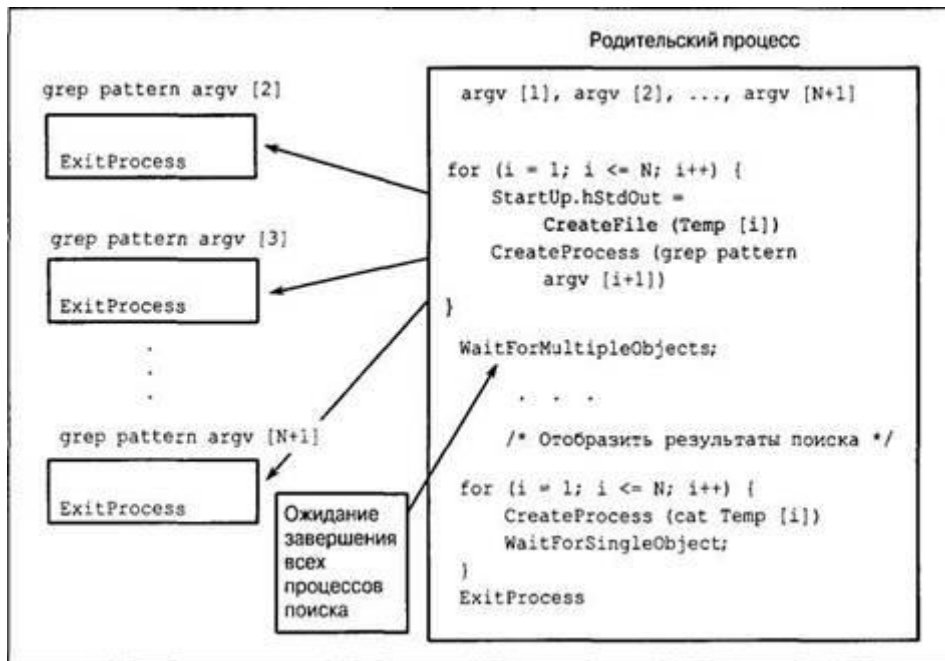


Рис. Поиск текстового шаблона в файлах с использованием нескольких процессов

Программа 8. grepMP: выполнение параллельного поиска текстового шаблона

/\* Глава 6. grepMP. \*/

/\* Версия команды grep, использующая несколько процессов. \*/

#include "EvryThng.h"

int \_tmain(DWORD argc, LPTSTR argv[])

/\* Для выполнения поиска в каждом из файлов, указанных в командной строке, создается отдельный процесс. Каждому процессу предоставляется временный файл в текущем каталоге, в котором сохраняются результаты. \*/

{

HANDLE hTempFile;

SECURITY\_ATTRIBUTES StdOutSA = /\* Атрибуты защиты для наследуемого дескриптора. \*/

{sizeof(SECURITY\_ATTRIBUTES), NULL, TRUE};

TCHAR CommandLine[MAX\_PATH + 100];

STARTUPINFO StartUpSearch, Startup;

PROCESS\_INFORMATION ProcessInfo;

DWORD iProc, ExCode;

HANDLE \*hProc; /\* Указатель на массив дескрипторов процессов. \*/

typedef struct {TCHAR TempFile[MAX\_PATH];} PROCFILE;

PROCFILE \*ProcFile; /\* Указатель на массив имен временных файлов. \*/

GetStartupInfo(&StartUpSearch);

GetStartupInfo(&Startup);

ProcFile = malloc((argc - 2) \* sizeof(PROCFILE));

hProc = malloc((argc - 2) \* sizeof(HANDLE));

/\* Создать для каждого файла отдельный процесс "grep". \*/

```

for (iProc = 0; iProc < argc - 2; iProc++) {
    _stprintf(CommandLine, _T("%s%s %s"), _T("grep "), argv[1], argv[iProc + 2]);
    GetTempFileName(_T("."), _T("gtm"), 0, ProcFile[iProc].TempFile); /* Для
хранения результатов поиска. */
    hTempFile = /* Этот дескриптор является наследуемым */
        CreateFile(ProcFile[iProc].TempFile, GENERIC_WRITE,
FILE_SHARE_READ | FILE_SHARE_WRITE, &StdOutSA,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    StartUpSearch.dwFlags = STARTF_USESTDHANDLES;
    StartUpSearch.hStdOutput = hTempFile;
    StartUpSearch.hStdError = hTempFile;
    StartUpSearch.hStdInput = GetStdHandle(STD_INPUT_HANDLE);
    /* Создать процесс для выполнения командной строки. */
    CreateProcess(NULL, CommandLine, NULL, NULL, TRUE, 0, NULL, NULL,
&StartUpSearch, &ProcessInfo);
    /* Закрыть ненужные дескрипторы. */
    CloseHandle(hTempFile);
    CloseHandle(ProcessInfo.hThread);
    hProc[iProc] = ProcessInfo.hProcess;
}
/* Выполнить все процессы и дождаться завершения каждого из них. */
for (iProc = 0; iProc < argc - 2; iProc += MAXIMUM_WAIT_OBJECTS)
WaitForMultipleObjects( /* Разрешить использование достаточно большого
количества процессов */
    min(MAXIMUM_WAIT_OBJECTS, argc - 2 - iProc), &hProc [iProc], TRUE,
INFINITE);
/* Переслать результирующие файлы на стандартный вывод с
использованием утилиты cat */
for (iProc = 0; iProc < argc - 2; iProc++) {
    if (GetExitCodeProcess(hProc[iProc], &ExCode) && ExCode==0) {
        /* Обнаружен шаблон — Вывести результаты. */
        if (argc > 3) _tprintf(_T("%s:\n"), argv [iProc + 2]);
        fflush(stdout); /* Использование стандартного вывода несколькими
процессами. */
        _stprintf(CommandLine, _T("%s%s"), _T("cat "), ProcFile[iProc].TempFile);
        CreateProcess(NULL, CommandLine, NULL, NULL, TRUE, 0, NULL, NULL,
&StartUp, &ProcessInfo);
        WaitForSingleObject(ProcessInfo.hProcess, INFINITE);
        CloseHandle(ProcessInfo.hProcess);
        CloseHandle(ProcessInfo.hThread);
    }
    CloseHandle(hProc [iProc]);
    DeleteFile(ProcFile[iProc].TempFile);
}
free(ProcFile);

```

```
free(hProc);
return 0;
}
```

**Задание 2.** Воспроизведите текст программы. Выполните его.

**Задание 3.** Подготовить отчет о выполненной работе

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

**Практическое занятие № 10**

**Тема раздела:** Библиотеки DLL

**Тема практической работы:** Создание библиотеки DLL

**Цель:** ознакомиться с созданием библиотеки DLL.

**Планируемые результаты:**

**уметь:** работать создавать библиотеки DLL.

**знать:** понятие "библиотека DLL".

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.**

Ознакомиться с теоретическими положениями

**Пошаговое руководство. Создание и использование собственной библиотеки динамической компоновки (**

<https://docs.microsoft.com/ru-ru/cpp/build/walkthrough-creating-and-using-a-dynamic-link-library-cpp?view=msvc-170> )

**Задание 2.** Создайте библиотеку, используя в качестве наполнения ранее созданные программы.

**Задание 3.** Подготовить отчет о выполненной работе

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## **Практическое занятие № 11, 12**

**Тема раздела:** Потоки и планирование выполнения

**Тема практической работы 11:** Использование потоков

**Тема практической работы 12:** Синхронизация потоков

**Цель:** ознакомиться с управлением потоками.

**Планируемые результаты:**

**уметь:** использовать и синхронизировать потоки

**знать:** понятие потока

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; задание; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Ознакомиться с теоретическими положениями

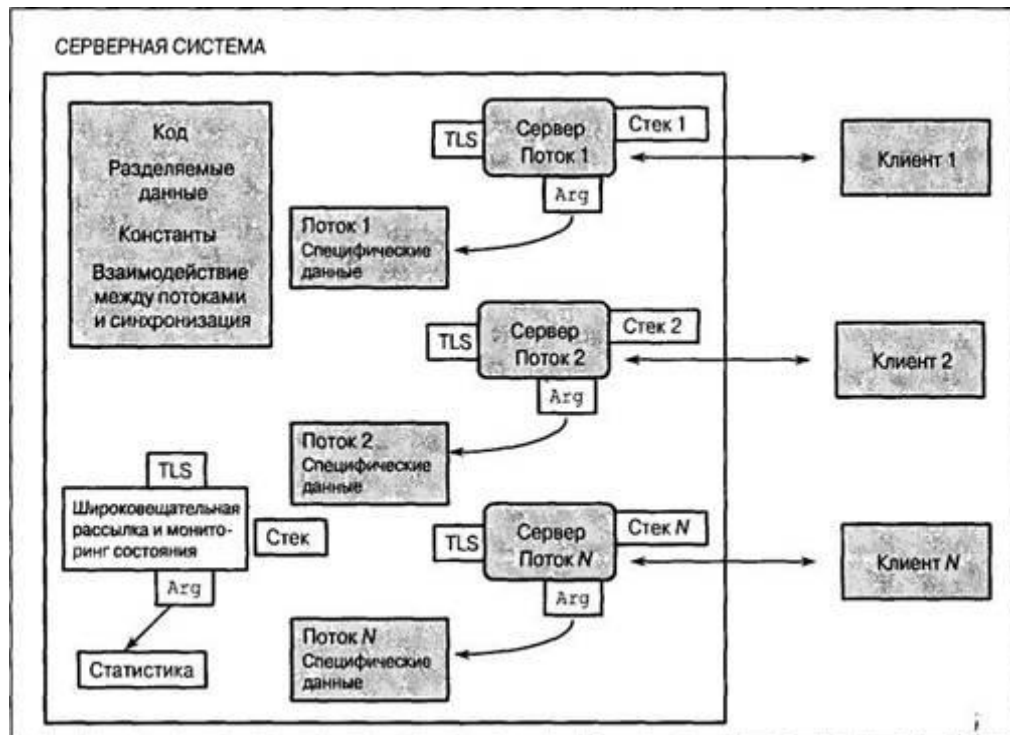
Книга: Системное программирование в среде Windows

### **Основные сведения о потоках**

Потоки, принадлежащие одному процессу, разделяют общие данные и код, поэтому очень важно, чтобы каждый поток имел также собственную область памяти, относящуюся только к нему. В Windows удовлетворение этого требования обеспечивается несколькими способами.

- У каждого потока имеется собственный стек, который она использует при вызове функций и обработке некоторых данных.
- При создании потока вызывающий процесс может передать ему аргумент (Arg на рис. Потоки в среде сервера), который обычно является указателем. На практике этот аргумент помещается в стек потока.
- Каждый поток может распределять индексы собственных локальных областей хранения (Thread Local Storage, TLS), а также считывать и устанавливать значения TLS. TLS, описанные далее, предоставляют в распоряжение потоков небольшие массивы данных, и каждый из потоков может обращаться к собственной TLS. Одним из преимуществ TLS является то, что они обеспечивают защиту данных, принадлежащих одному потоку, от воздействия со стороны других потоков.





**Рис.** Потоки в среде сервера

Аргумент потока и TLS могут использоваться для указания произвольной структуры данных. Применительно к представленному на рис. Потоки в среде сервера примеру сервера эта структура может содержать текущий запрос и отклик потока на этот запрос, а также предоставлять рабочую память для других целей.

В случае SMP-систем Windows обеспечивает параллельное выполнение различных потоков, в том числе и принадлежащих одному и тому же процессу, на разных процессорах. Правильное использование этой возможности позволяет повысить производительность, однако, как будет показано в двух следующих главах, в результате непродуманных действий без заранее определенной стратегии использования нескольких процессоров производительность SMP-систем может даже ухудшиться по сравнению с однопроцессорными системами.

Вероятно, вы не будете удивлены, узнав о том, что у потоков, как и у любого другого объекта Windows, имеются дескрипторы и что для создания потоков, выполняющихся в адресном пространстве вызывающего процесса, предусмотрен системный вызов `CreateThread`. Как и в случае процессов, мы будем говорить иногда о "родительских" и "дочерних" потоках, хотя ОС не делает в этом отношении никаких различий. Системный вызов `CreateThread` предъявляет ряд специфических требований:

- Укажите начальный адрес потока в коде процесса.
- Укажите размер стека, и необходимое пространство стека будет выделено из виртуального адресного пространства процесса. Размер стека по умолчанию равен размеру стека основного потока (обычно 1 Мбайт). Первоначально для стека отводится одна страница (см. главу 5). Новые

страницы стека выделяются по мере надобности до тех пор, пока стек не достигнет своего максимального размера, поэтому не сможет больше расти.

- Задайте указатель на аргумент, передаваемый потоку. Этот аргумент может быть чем угодно и должен интерпретироваться самим потоком.

- Функция возвращает значение идентификатора (ID) и дескриптор потока.

В случае ошибки возвращаемое значение равно NULL.

*HANDLE CreateThread(LPSECURITY\_ATTRIBUTES lpsa, DWORD dwStackSize, LPTHREAD\_START\_ROUTINE lpStartAddr, LPVOID lpThreadParm, DWORD dwCreationFlags, LPDWORD lpThreadId)*

Параметры

*lpsa* — указатель на уже хорошо знакомую структуру атрибутов защиты.

*dwStackSize* — размер стека нового потока в байтах. Значению 0 этого параметра соответствует размер стека по умолчанию, равный размеру стека основного потока.

*lpStartAddr* — указатель на функцию (принадлежащую контексту процесса), которая должна выполняться. Эта функция принимает единственный аргумент в виде указателя и возвращает 32-битовый код завершения. Этот аргумент может интерпретироваться потоком либо как переменная типа *DWORD*, либо как указатель. Функция потока (*ThreadFunc*) имеет следующую сигнатуру:

*DWORD WINAPI ThreadFunc(LPVOID)*

*lpThreadParm* — указатель, передаваемый потоку в качестве аргумента, который обычно интерпретируется потоком как указатель на структуру аргумента.

*dwCreationFlags* — если значение этого параметра установлено равным 0, то поток запускается сразу же после вызова функции *CreateThread*. Установка значения *CREATE\_SUSPENDED* приведет к запуску потока в приостановленном состоянии, из которого поток может быть переведен в состояние готовности путем вызова функции *ResumeThread*.

*lpThreadId* — указатель на переменную типа *DWORD*, которая получает идентификатор нового потока; в Windows 9x и Windows NT 3.51 значение NULL для этого параметра устанавливать нельзя.

Любой поток процесса может сама завершить свое выполнение, вызвав функцию *ExitThread*, однако более обычным способом самостоятельного завершения потока является возврата из функции потока с использованием кода завершения в качестве возвращаемого значения. По завершении выполнения потока память, занимаемая ее стеком, освобождается. В случае если поток был создан в библиотеке DLL, будет вызвана соответствующая точка входа *DllMain* (глава 4) с указанием флага *DLL\_THREAD\_DETACH* в качестве "причины" этого вызова.

*VOID ExitThread(DWORD dwExitCode)*

Когда завершается выполнение последнего потока, завершается и выполнение самого процесса.

Выполнение потока также может быть завершено другим потоком с помощью функции `TerminateThread`, однако освобождения ресурсов потока при этом не происходит, обработчики завершения не выполняются и уведомления библиотекам DLL не посылаются. Лучше всего, когда поток сам завершает свое выполнение; применять для этого функцию `TerminateThread` крайне нежелательно. Функции `TerminateThread` присущи те же недостатки, что и функции `TerminateProcess`.

Поток, выполнение которого было завершено (напомним, что обычно поток должен самостоятельно завершать свое выполнение), продолжает существовать до тех пор, пока посредством функции `CloseHandle` не будет закрыт ее последний дескриптор. Любой другой поток, возможно и такой, который ожидает завершения другого потока, может получить код завершения потока.

***BOOL GetExitCodeThread(HANDLE hThread, LPDWORD lpExitCode)***

`lpExitCode` — будет содержать код завершения потока, указывающий на его состояние. Если поток еще не завершен, значение этой переменной будет равно `STILL_ACTIVE`.

### **Идентификация потоков**

Функции, используемые для получения идентификаторов (ID) и дескрипторов потоков, напоминают те, которые используются для аналогичных целей в случае процессов.

- `GetCurrentThread` — возвращает ненаследуемый псевдодескриптор вызывающего потока.
- `GetCurrentThreadId` — позволяет получить идентификатор потока, а не его дескриптор.
- `GetThreadId` — позволяет получить идентификатор потока, если известен его дескриптор; эта функция требует использования Windows Server 2003.
- `OpenThread` — создает дескриптор потока по известному идентификатору.

В программе `JobShell` (программа 6.3) нам очень пригодилась функция `OpenProcess`, и функция `OpenThread` может применяться для аналогичных целей.

### **Дополнительные функции управления потоками**

Несмотря на то что функций управления потоками, которые мы выше обсуждали, вполне достаточно для большинства случаев, в том числе и для примеров, в Windows XP и Windows Server 2003 были введены две дополнительные функции. Их краткие описания представлены ниже.

1. Функция `GetProcessIdOfThread`, требующая использования Windows Server 2003, позволяет получать идентификатор процесса, которому принадлежит поток, по известному дескриптору потока. Вы могли бы задействовать эту функцию в программах, предназначенных для управления потоками, принадлежащими другим процессам, или взаимодействия с такими потоками. Если необходимо получить дескриптор процесса, применяйте для этого функцию `OpenProcess`.

2. Функция `GetThreadIOPendingFlag` позволяет определить, имеются ли у потока, на который указывает дескриптор, необслуженные запросы ввода/вывода. Например, поток мог быть заблокирован во время выполнения операции `ReadFile`. В качестве результата возвращается состояние потока во время выполнения данной функции; фактическое состояние может в любой момент измениться, если целевой поток завершает или начинает выполнение операции. Эта функция требует использования NT 5.1 и поэтому доступна лишь в Windows XP или Windows Server 2003.

### **Приостановка и возобновление выполнения потока**

Для каждого потока поддерживается счетчик приостановок (`suspend count`), и выполнение потока может быть продолжено лишь в том случае, если значение этого счетчика равно 0. Поток может увеличивать или уменьшать значение счетчика приостановок другого потока с помощью функций `SuspendThread` и `ResumeThread`. Вспомните, что поток можно создать в приостановленном состоянии со счетчиком приостановок равным 1.

*`DWORD ResumeThread(HANDLE hThread)`*

*`DWORD SuspendThread(HANDLE hThread)`*

В случае успешного выполнения обе функции возвращают предыдущее значение счетчика приостановок, иначе — `0xFFFFFFFF`.

### **Ожидание завершения потока**

Поток может дожидаться завершения выполнения другого потока точно так же, как потоки могут дожидаться завершения процесса, что обсуждалось в главе 6. В этом случае при вызове функций ожидания (`WaitForSingleObject` и `WaitForMultipleObjects`) вместо дескрипторов процессов следует использовать дескрипторы потоков. Заметьте, что не все дескрипторы в массиве, передаваемом функции `WaitForMultipleObjects`, должны быть обязательно одного и того же типа; например, в одном вызове могут быть одновременно указаны дескрипторы потоков, процессов и других объектов.

Допустимое количество объектов, одновременно ожидаемых функцией `WaitForMultipleObjects`, ограничено значением `MAXIMUM_WAIT_OBJECTS`, но при большом количестве потоков можно воспользоваться серией вызовов функций ожидания. Эта техника уже была продемонстрирована в программе 9; программы, приведенные в книге, ожидают завершения выполнения одиночных объектов, но на Web-сайте приведены полные решения.

Функция ожидания дожидается, пока объект, указанный дескриптором, не перейдет в сигнальное состояние. В случае потоков объект потока переводится в сигнальное состояние при помощи функций `ExitThread` и `TerminateThread`, что приводит к освобождению всех других потоков, ожидающих перехода данного объекта в сигнальное состояние, включая и те потоки, которые могли оставаться в состоянии ожидания и впоследствии, после того, как поток завершится. Дескриптор потока, перешедший в сигнальное состояние, не выходит из этого состояния. То же самое остается

справедливым и по отношению к дескрипторам процессов, но не относится к дескрипторам некоторых других объектов, например, мьютексов и событий.

Заметьте, что дожидаться перехода в сигнальное состояние одного и того же объекта могут одновременно несколько потоков. Аналогично, функция `ExitProcess` переводит в сигнальное состояние как сам процесс, так и все его потоки.

### **Удаленные потоки**

Функция `CreateRemoteThread` позволяет создавать потоки, выполняющиеся в другом процессе. По сравнению с функцией `CreateThread` в ней имеется один дополнительный параметр для указания дескриптора процесса, а адрес функции, задающий начальный адрес нового потока, должен находиться в адресном пространстве целевого процесса. Использование функции `CreateRemoteThread` относится к числу интересных, однако рискованных способов непосредственного воздействия одним процессом на другой, и может пригодиться, например, при написании отладчиков.

У функции `CreateRemoteThread` есть одно очень интересное применение. Вместо того чтобы вызывать функцию `TerminateProcess`, управляющий процесс может создать поток, выполняющийся в другом процессе, который и организует корректное завершение этого процесса.

*Понятие о потоках твердо упрочилось во многих ОС, и исторически так сложилось, что многие поставщики и пользователи UNIX предоставляли собственные частные варианты их реализации. Были разработаны некоторые библиотеки, обеспечивающие многопоточную поддержку вне ядра. В настоящее время стандартом в этой области являются потоки POSIX Pthreads. Потоки Pthreads включены в частные варианты реализации UNIX и Linux и иногда считаются частью UNIX. Соответствующие системные вызовы отличаются от обычных системных вызовов UNIX наличием в именах префикса pthread. Потоки Pthreads поддерживаются также некоторыми другими системами, отличными от UNIX, такими, например, как Open VMS.*

*Системный вызов `pthread_create` эквивалентен вызову `CreateThread`, а системный вызов `pthread_exit` — вызову `ExitThread`. Для организации ожидания одним потоком завершения другого применяется системный вызов `pthread_join`. Потоки Pthreads предоставляют очень полезную функцию `pthread_cancel`, гарантирующую, в отличие от функции `TerminateThread`, выполнение обработчиков завершения и уничтожение ненужных дескрипторов. Возможность уничтожения потоков была бы в Windows крайне желательной, но в главе 10 представлен метод, обеспечивающий получение такого же эффекта.*

### **Использование библиотеки C в потоках**

В большинстве программ требуется библиотека C, хотя бы для того, чтобы обеспечить выполнение операций над строками. Исторически так сложилось, что библиотека C была рассчитана на применение в однопоточных процессах, поэтому для хранения промежуточных результатов

многие функции используют области глобальной памяти. Подобные библиотеки, в которых отсутствует многопоточная поддержка, не являются безопасными (thread-safe) с точки зрения одновременного выполнения нескольких потоков, поскольку, например, одновременно две независимые потоки могут пытаться получить доступ к библиотеке и изменить данные, содержащиеся в ее глобальной памяти.

Пример функции strtok показывает, почему при написании некоторых функций библиотеки C не учитывалась многопоточная поддержка. Функция strtok, просматривающая строку в поиске очередного вхождения определенной лексемы, поддерживает сохранение состояния (persistent state) между последовательными вызовами функции, и это состояние хранится в области статической памяти, совместный доступ к которой имеют все потоки, вызывающие эту функцию.

Microsoft C решает эту проблему, предлагая реализацию библиотеки C под названием LIBCMT.LIB, которая обеспечивает многопоточную поддержку. Однако, это еще не все. Вы не должны использовать функцию CreateThread; для запуска потока и создания специфической для него области рабочей памяти библиотеки LIBCMT.LIB необходимо пользоваться специальной функцией C, а именно, функцией \_beginthreadex. Для завершения потока вместо функции ExitThread применяется функция \_endthreadex.

#### **Примечание**

*В качестве упрощенного варианта функции \_beginthreadex предусмотрена функция \_beginthread, однако использовать ее не рекомендуется. Прежде всего, функция \_beginthread не имеет ни атрибутов, ни флагов защиты и не возвращает идентификатор потока. Более того, в действительности она закрывает дескриптор потока, который создает, в результате чего возвращенное значение дескриптора может оказаться недействительным на момент его сохранения родительским потоком. Не следует вызывать и функцию \_endthread; она не позволяет пользоваться возвращаемым значением.*

Аргументы функции \_beginthreadex в точности совпадают с аргументами функций Windows, однако типы данных Windows для этой функции не определены, и поэтому тип возвращаемого значения функции \_beginthread необходимо привести к типу HANDLE, что позволит избежать появления предупреждающих сообщений. Убедитесь в том, что определение символической константы \_MT предшествует любому из включаемых файлов; в примерах программ это определение содержится в файле Envirmnt.h. Больше от вас ничего не требуется. Резюмируя, перечислим действия, которые вы должны выполнить, если имеете дело со средой разработки Visual C++.

- Подключите библиотеку LIBCMT.LIB и откажитесь от использования библиотеки, заданной по умолчанию.
- Включите директиву #define \_MT во все исходные файлы, в которых используется библиотека C.

- Добавьте включаемый файл `<process.h>`, содержащий определения функций `_beginthreadex` и `_endthreadex`.
- Создайте потоки с помощью функции `_beginthreadex`; не применяйте для этой цели функцию `CreateThread`.
- Завершите потоки посредством функции `_endthreadex` или просто воспользуйтесь оператором `return` в конце функции потока.

В приложении А вы найдете указания относительно того, как создавать многопоточные приложения. В частности, можно, и даже рекомендуется, указывать библиотеку и определять константу `_MT` непосредственно в среде разработки.

Именно так будут построены все наши примеры, и функция `CreateThread` никогда не будет непосредственно применяться в программах даже в тех случаях, когда библиотека `C` в функциях потоков не используется.

### **Пример: многопоточный поиск контекста**

В программе 9 (`grepMP`) для выполнения одновременного поиска текстового шаблона в нескольких файлах использовались процессы. Программа 10 (`grepMT`), которая включает исходный код функции поиска текстового шаблона `grep`, обеспечивает выполнение поиска несколькими потоками в рамках одного процесса. Код функции поиска основан на вызовах функций файлового ввода/вывода библиотеки `C`. Основная программа аналогична той, которая предлагалась в варианте реализации, основанном на использовании процессов.

Этот пример также показывает, что применение потоков позволяет выполнять асинхронные операции ввода/вывода даже без привлечения специально для этого предназначенных методов. В данном примере параллельным вводом/выводом с участием нескольких файлов управляет программа, в то время как основной или любого другого потока предоставляется возможность в ожидании завершения ввода/вывода выполнять дополнительную обработку. По мнению автора, способ реализации асинхронного ввода/вывода, обеспечиваемый потоками, является более простым, поможет вам выработать собственное мнение на этот счет.

Мы увидим, однако, что в сочетании с портами завершения ввода/вывода операции асинхронного ввода/вывода становятся очень полезным, а часто и необходимым средством в тех случаях, когда количество потоков очень велико.

В иллюстративных целях в программу `grepMT` введено дополнительное отличие по сравнению с программой `grepMP`. В данном случае функция `WaitForMultipleObjects` ожидает завершения не всех потоков, а только одного. Соответствующая информация выводится без ожидания завершения других потоков. В большинстве случаев порядок завершения потоков будет меняться от одного запуска программы к другому. Программу легко видоизменить таким образом, чтобы результаты отображались в порядке указания аргументов в командной строке; для этого будет достаточно симитировать программу `grepMP`.

Наконец, обратите внимание на ограничение в 64 потока, обусловленное значением константы `MAXIMUM_WAIT_OBJECTS`, которая ограничивает количество дескрипторов при вызове функции `WaitForMultipleObjects`. Если у вас возникнет необходимость в большем количестве потоков, организуйте для функций `WaitForSingleObjects` или `WaitForMultipleObjects` соответствующий цикл.

### ***Предостережение***

*Программа `grepMP` осуществляет асинхронный ввод/вывод в том смысле, что отдельные потоки выполняют параллельное синхронное чтение различных файлов, которые блокируются до момента завершения операции чтения. Можно также организовать параллельное чтение одного и того же файла, если у него имеются различные дескрипторы (обычно, по одному дескриптору для каждого потока). Эти дескрипторы должны быть сгенерированы функцией `CreateFile`, а не функцией `DuplicateHandle`. В главе 14 описывается асинхронный ввод/вывод, осуществляемый как с использованием, так и без использования пользовательских потоков, а в примере, доступном на Web-сайте (программа `atouMT`, описанная в главе 14), операции ввода/вывода выполняются с использованием нескольких потоков по отношению к одному и тому же файлу.*

### **Пример: параллельный поиск указанного текстового шаблона**

Настало время посмотреть на процессы Windows в действии. Приведенная ниже в качестве примера программа `grepMP` создает процессы для поиска указанного текстового шаблона в файлах, по одному процессу на каждый файл. Эта программа моделирует UNIX-утилиту `grep`, хотя используемая нами методика применима к любой программе, которая полагается на стандартный вывод. Рассматривайте программу поиска как "черный ящик" и считайте, что она является просто исполняемой программой, выполнение которой должно контролироваться родительским процессом.

Командная строка программы имеет следующий вид:

`grepMP шаблон F1 F2 ... FN`

Программа 9 выполняет следующие виды обработки:

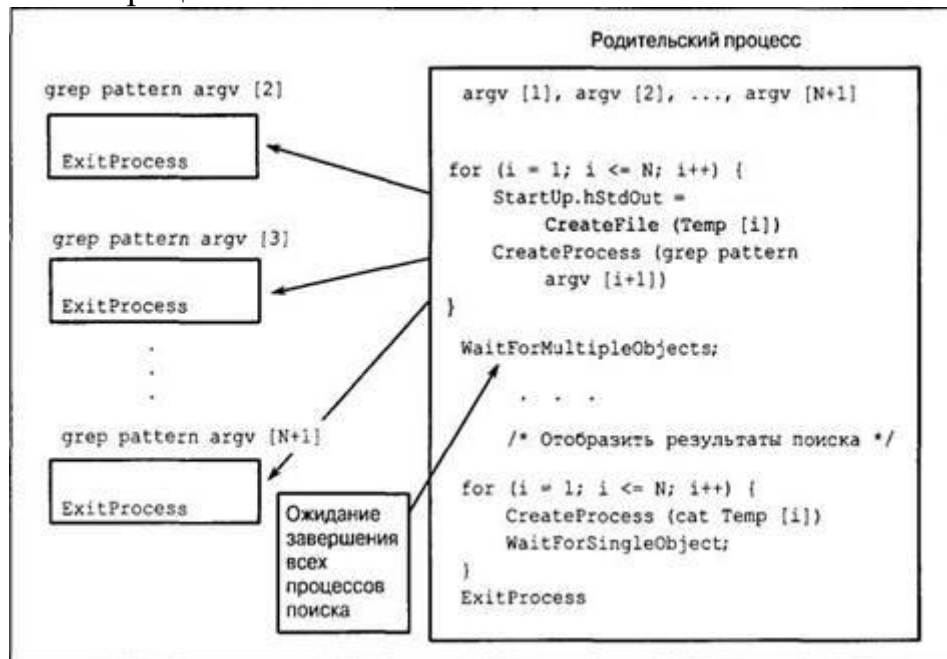
- Для поиска указанного шаблона в каждом из входных файлов, от `F1` до `FN`, используется отдельный процесс, запускающий один и тот же исполняемый модуль. Для каждого процесса программа создает командную строку такого вида: `grep шаблон FK`.
- Полю `hStdOut` структуры `STARTUPINFO` нового процесса присваивается значение дескриптора временного файла, который определяется как наследуемый.
- Программа организует ожидание завершения всех процессов поиска, используя для этого функцию `WaitForMultipleObjects`.
- По завершении всех процессов поиска осуществляется поочередный вывод результатов (временных файлов). Вывод временного файла осуществляет процесс, выполняющий утилиту `cat`.



- Возможности функции WaitForMultipleObjects ограничиваются лишь максимально допустимым количеством дескрипторов, которое устанавливается значением MAXIMUM\_WAIT\_OBJECTS (64), поэтому она вызывается многократно.

- Для определения успешности попытки нахождения данным процессом заданного шаблона программа использует код завершения процесса grep.

Порядок обработки файлов программой 9 иллюстрируется на рис. Поиск текстового шаблона в файлах с использованием нескольких процессов.



**Рис.** Поиск текстового шаблона в файлах с использованием нескольких процессов

Программа 9. grepMP: выполнение параллельного поиска текстового шаблона

```

/* Глава 6. grepMP. */
/* Версия команды grep, использующая несколько процессов. */
#include "EvryThng.h"
int _tmain(DWORD argc, LPTSTR argv[])
/* Для выполнения поиска в каждом из файлов, указанных в командной
строке, создается отдельный процесс. Каждому процессу предоставляется
временный файл в текущем каталоге, в котором сохраняются результаты. */
{
    HANDLE hTempFile;
    SECURITY_ATTRIBUTES StdOutSA = /* Атрибуты защиты для
наследуемого дескриптора. */
        {sizeof(SECURITY_ATTRIBUTES), NULL, TRUE};
    TCHAR CommandLine[MAX_PATH + 100];
    STARTUPINFO StartUpSearch, Startup;
  
```

```

PROCESS_INFORMATION ProcessInfo;
DWORD iProc, ExCode;
HANDLE *hProc; /* Указатель на массив дескрипторов процессов. */
typedef struct {TCHAR TempFile[MAX_PATH];} PROCFILE;
PROCFILE *ProcFile; /* Указатель на массив имен временных файлов.
*/

GetStartupInfo(&StartUpSearch);
GetStartupInfo(&StartUp);
ProcFile = malloc((argc - 2) * sizeof(PROCFILE));
hProc = malloc((argc - 2) * sizeof(HANDLE));
/* Создать для каждого файла отдельный процесс "grep". */
for (iProc = 0; iProc < argc - 2; iProc++) {
    _sprintf(CommandLine, _T("%s%s %s"), _T("grep "), argv[1], argv[iProc
+ 2]);
    GetTempFileName(_T("."), _T("gtm"), 0, ProcFile[iProc].TempFile); /*
Для хранения результатов поиска.*/
    hTempFile = /* Этот дескриптор является наследуемым */
        CreateFile(ProcFile[iProc].TempFile, GENERIC_WRITE,
FILE_SHARE_READ | FILE_SHARE_WRITE, &StdOutSA,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    StartUpSearch.dwFlags = STARTF_USESTDHANDLES;
    StartUpSearch.hStdOutput = hTempFile;
    StartUpSearch.hStdError = hTempFile;
    StartUpSearch.hStdInput = GetStdHandle(STD_INPUT_HANDLE);
    /* Создать процесс для выполнения командной строки. */
    CreateProcess(NULL, CommandLine, NULL, NULL, TRUE, 0, NULL,
NULL, &StartUpSearch, &ProcessInfo);
    /* Закрыть ненужные дескрипторы. */
    CloseHandle(hTempFile);
    CloseHandle(ProcessInfo.hThread);
    hProc[iProc] = ProcessInfo.hProcess;
}
/* Выполнить все процессы и дождаться завершения каждого из них. */
for (iProc = 0; iProc < argc - 2; iProc += MAXIMUM_WAIT_OBJECTS)
WaitForMultipleObjects( /* Разрешить использование достаточно большого
количества процессов */
    min(MAXIMUM_WAIT_OBJECTS, argc - 2 - iProc), &hProc [iProc],
TRUE, INFINITE);
/* Переслать результирующие файлы на стандартный вывод с
использованием утилиты cat */
for (iProc = 0; iProc < argc - 2; iProc++) {
    if (GetExitCodeProcess(hProc[iProc], &ExCode) && ExCode==0) {
        /* Обнаружен шаблон — Вывести результаты. */
        if (argc > 3) _tprintf(_T("%s:\n"), argv [iProc + 2]);
    }
}

```

```

        fflush(stdout); /* Использование стандартного вывода несколькими
процессами. */
        _stprintf(CommandLine, _T("%s%s"), _T("cat "),
ProcFile[iProc].TempFile);
        CreateProcess(NULL, CommandLine, NULL, NULL, TRUE, 0, NULL,
NULL, &Startup, &ProcessInfo);
        WaitForSingleObject(ProcessInfo.hProcess, INFINITE);
        CloseHandle(ProcessInfo.hProcess);
        CloseHandle(ProcessInfo.hThread);
    }
    CloseHandle(hProc [iProc]);
    DeleteFile(ProcFile[iProc].TempFile);
}
free(ProcFile);
free(hProc);
return 0;
}

```

Программа 10. grepMT: многопоточный поиск текстового шаблона

/\* Глава 7. grepMT. \*/

/\* Параллельный поиск текстового шаблона — версия, использующая несколько потоков. \*/

```

#include "EvryThng.h"
typedef struct { /* Структура данных потока поиска. */
    int argc;
    TCHAR targv[4][MAX_PATH];
} GREP_THREAD_ARG;
typedef GREP_THREAD_ARG *PGR_ARGS;
static DWORD WINAPI ThGrep(PGR_ARGS pArgs);
int _tmain(int argc, LPTSTR argv[]) {
    GREP_THREAD_ARG * gArg;
    HANDLE * tHandle;
    DWORD ThdIdxP, ThId, ExitCode;
    TCHAR CmdLine[MAX_COMMAND_LINE];
    int iThrd, ThdCnt;
    STARTUPINFO Startup;
    PROCESS_INFORMATION ProcessInfo;
    GetStartupInfo(&Startup);
    /* Основной поток: создает отдельные потоки поиска на основе
функции "grep" для каждого файла. */
    tHandle = malloc((argc - 2) * sizeof(HANDLE));
    gArg = malloc((argc - 2) * sizeof(GREP_THREAD_ARG));
    for (iThrd = 0; iThrd < argc - 2; iThrd++) {
        _tcscpy(gArg[iThrd].targv[1], argv[1]); /* Pattern. */
        _tcscpy(gArg[iThrd].targv[2], argv[iThrd + 2]);
    }
}

```

```

    GetTempFileName /* Имя временного файла. */
    (".", "Gre", 0, gArg[iThrd].targv[3]);
    gArg[iThrd].argc = 4;
    /* Создать рабочий поток для выполнения командной строки. */
    tHandle[iThrd] = (HANDLE)_beginthreadex(NULL, 0, ThGrep,
&gArg[iThrd], 0, &ThId);
}
/* Перенаправить стандартный вывод для вывода списка файлов. */
Startup.dwFlags = STARTF_USESTDHANDLES;
Startup.hStdOutput = GetStdHandle(STD_OUTPUT_HANDLE);
/* Выполняются все рабочие потоки. Ожидать их завершения. */
ThdCnt = argc - 2;
while (ThdCnt > 0) {
    ThdIdxP = WaitForMultipleObjects(ThdCnt, tHandle, FALSE, INFINITE);
    iThrd = (int)ThdIdxP - (int)WAIT_OBJECT_0;
    GetExitCodeThread(tHandle [iThrd], &ExitCode);
    CloseHandle(tHandle [iThrd]);
    if (ExitCode ==0) { /* Шаблон найден. */
        if (argc > 3) {
            /* Вывести имя файла, если имеется несколько файлов. */
            _tprintf(_T("\n**Результаты поиска – файл: %s\n"), gArg[iThrd].targv
[2]);
            fflush(stdout);
        }
        /* Использовать программу "cat" для перечисления результирующих
файлов. */
        _stprintf(CmdLine, _T("%s%s"), _T("cat "), gArg [iThrd].targv[3]);
        CreateProcess(NULL, CmdLine, NULL, NULL, TRUE, 0, NULL, NULL,
&StartUp, &ProcessInfo);
        WaitForSingleObject(ProcessInfo.hProcess, INFINITE);
        CloseHandle(ProcessInfo.hProcess);
        CloseHandle(ProcessInfo.hThread);
    }
    DeleteFile(gArg[iThrd].targv[3]);
    /* Скорректировать массивы потоков и имен файлов. */
    tHandle[iThrd] = tHandle[ThdCnt - 1];
    _tcscpy(gArg[iThrd].targv[3], gArg[ThdCnt - 1].targv[3]);
    _tcscpy(gArg[iThrd].targv[2], gArg[ThdCnt - 1].targv[2]);
    ThdCnt--;
}
}
/* Прототип функции контекстного поиска:
static DWORD WINAPI ThGrep(PGR_ARGS pArgs){ } */

```

## Потоки и производительность

Программы `grepMP` и `grepMT` по своей структуре и сложности сопоставимы друг с другом, однако, как и следовало ожидать, программа `grepMT` характеризуется более высокой производительностью, так как переключение между потоками осуществляется ядром намного эффективнее, чем переключение между процессами. В приложении В показано, что эти теоретические ожидания отвечают действительности, и это особенно заметно в тех случаях, когда файлы размещены на различных дисках. Оба варианта реализации способны работать в SMP-системах, существенно улучшая показатели производительности в терминах общего времени выполнения (истекшего времени); потоки, независимо от того, принадлежат ли они одному и тому же или разным процессам, параллельно выполняются на различных процессорах. Измеренное пользовательское время в действительности превышает общее время выполнения, поскольку рассчитывается в виде суммарной величины для всех процессоров.

В то же время, существует весьма распространенное заблуждение, суть которого состоит в том, что отмеченный параллелизм, независимо от того, касается ли он использования нескольких процессов, как в случае `grepMP`, или же применения нескольких потоков, как в случае `grepMT`, способен приводить к повышению производительности лишь в случае SMP-систем. Выигрыш в производительности можно получить и при использовании нескольких дисков, а также при любом другом распараллеливании в системе хранения. Во всех подобных случаях операции ввода/вывода с участием нескольких файлов будут осуществляться в параллельном режиме.

### **Пример: применение принципа "разделяй и властвуй" для решения задачи сортировки слиянием в SMP-системах**

Этот пример демонстрирует возможности значительного повышения производительности за счет использования потоков, особенно в случае SMP-систем. Основная идея заключается в разбиении задачи на более мелкие составляющие, распределении выполнения подзадач между отдельными потоками и последующем объединении результатов для получения окончательного решения. Планировщик Windows автоматически назначит потокам отдельные процессоры, в результате чего задачи будут выполняться параллельно, снижая общее время выполнения приложения.

Эта стратегия, которую часто называют стратегией "разделяй и властвуй" (divide and conquer), или моделью рабочей группы (work crew model), оказалась весьма полезной и в качестве средства повышения производительности, и в качестве метода проектирования алгоритмов. Одним из примеров ее применения служит программа `grepMT` (программа 10), в которой для каждой файловой операции ввода/вывода и для поиска шаблона создается отдельный поток. Как показано в приложении В, в случае SMP-систем производительность повышается, поскольку планировщик может распределять выполнение потоков между различными процессорами.

Далее мы рассмотрим другой пример, в котором задача сортировки содержимого файла разбивается на ряд подзадач, выполнение которых делегируется отдельным потокам.

Решение задачи сортировки слиянием (merge-sort), в которой сортируемый массив разбивается на несколько массивов меньшего размера, является классическим примером алгоритма, построенного на принципе "разделяй и властвуй". Каждый из массивов небольшого размера сортируется по отдельности, после чего отсортированные массивы попарно объединяются с образованием отсортированных массивов большего размера. Описанное слияние массивов попарно осуществляется вплоть до завершения всего процесса сортировки. В общем случае, сортировка слиянием начинается с массивов размерности 1, которые сами по себе не нуждаются в сортировке. В данном примере сортировка начинается с массивов большей размерности, чтобы на каждый процессор приходилось по одному массиву. Блок-схема используемого алгоритма показана на рис. 11.

Детали реализации представлены в программе 11. Число задач задается пользователем в командной строке. Временные показатели сортировки приведены в приложении В ([http://programming-lang.com/ru/comp\\_osnet/hart/0/j655.html](http://programming-lang.com/ru/comp_osnet/hart/0/j655.html)).

Заметьте, что эта программа эффективно выполняется в однопроцессорных системах, в которых имеется достаточно большой запас оперативной памяти, и обеспечивает значительное повышение производительности в SMP-системах. Предостережение. Представленный алгоритм будет корректно работать лишь при условии, что число записей в сортируемом файле нацело делится на число потоков, а число потоков выражается степенью 2.

### ***Примечание***

*Изучая работу этой программы, постарайтесь отделить логику управления потоками от логики определения части массива, которую должна сортировать тот или иной поток. Обратите также внимание на использование функции `qsort` из библиотеки C, применение которой избавляет нас от необходимости самостоятельно разрабатывать эффективную функцию сортировки.*

Программа 11. sortMT: сортировка слиянием с использованием нескольких потоков

/\* Глава 7. SortMT.

Сортировка файлов с использованием нескольких потоков (рабочая группа).

```
sortMT [параметры] число_задач файл */
#include "EvryThng.h"
#define DATALEN 56 /* Данные: 56 байт; ключ: 8 байт. */
#define KEYLEN 8
typedef struct _RECORD {
    CHAR Key[KEYLEN];
    TCHAR Data[DATALEN];
} RECORD;
#define RECSIZE sizeof (RECORD)
typedef RECORD * LPRECORD;
```

```

typedef struct _THREADARG { /* Аргумент потока */
    DWORD iTh; /* Номер потока: 0, 1, 2, ... */
    LPRECORD LowRec; /* Младшая часть указателя записи */
    LPRECORD HighRec; /* Старшая часть указателя записи */
} THREADARG, *PTHREADARG;
static int KeyCompare(LPCTSTR, LPCTSTR);
static DWORD WINAPI ThSort(PTHREADARG pThArg);
static DWORD nRec; /* Общее число записей, подлежащих сортировке.
*/

static HANDLE* ThreadHandle;
int _tmain(int argc, LPTSTR argv[]) {
    HANDLE hFile;
    LPRECORD pRecords = NULL;
    DWORD FsLow, nRead, LowRecNo, nRecTh, NPr, ThId, iTh;
    BOOL NoPrint;
    int iFF, iNP;
    PTHREADARG ThArg;
    LPTSTR StringEnd;
    iNP = Options(argc, argv, _T("n"), &NoPrint, NULL);
    iFF = iNP + 1;
    NPr = _ttoi(argv[iNP]); /* Количество потоков. */
    hFile = CreateFile(argv[iFF], GENERIC_READ | GENERIC_WRITE, 0,
NULL, OPEN_EXISTING, 0, NULL);
    FsLow = GetFileSize(hFile, NULL);
    nRec = FsLow / RECSIZE; /* Общее число записей. */
    nRecTh = nRec / NPr; /* Количество записей на один поток. */
    /* Распределить память для аргументов потока и массива дескрипторов
и выделить в памяти место для файла. Считать весь файл. */
    ThArg = malloc(NPr * sizeof(THREADARG));
    /* Аргументы потоков. */
    ThreadHandle = malloc(NPr * sizeof(HANDLE));
    pRecords = malloc(FsLow + sizeof(TCHAR));
    ReadFile(hFile, pRecords, FsLow, &nRead, NULL);
    CloseHandle(hFile);
    LowRecNo = 0; /* Создать потоки, выполняющие сортировку. */
    for (iTh = 0; iTh < NPr; iTh++) {
        ThArg[iTh].iTh = iTh;
        ThArg[iTh].LowRec = pRecords + LowRecNo;
        ThArg[iTh].HighRec = pRecords + (LowRecNo + nRecTh);
        LowRecNo += nRecTh;
        ThreadHandle[iTh] = (HANDLE)_beginthreadex (NULL, 0, ThSort,
&ThArg[iTh], CREATE_SUSPENDED, &ThId);
    }
    for (iTh = 0; iTh < NPr; iTh++) /* Запустить все потоки сортировки. */
        ResumeThread(ThreadHandle [iTh]);
}

```

```

WaitForSingleObject(ThreadHandle[0], INFINITE);
for (iTh = 0; iTh < NPr; iTh++) CloseHandle(ThreadHandle [iTh]);
StringEnd = (LPTSTR)pRecords + FsLow;
*StringEnd = '\0';
if (!NoPrint) printf("\n%s", (LPCTSTR)pRecords);
free(pRecords);
free(ThArg);
free(ThreadHandle);
return 0;
} /* Конец tmain. */

```

```

static VOID MergeArrays(LPRECORD, LPRECORD);
DWORD WINAPI ThSort(PTHREADARG pThArg) {
    DWORD GrpSize = 2, RecsInGrp, MyNumber, TwoToI = 1;
    LPRECORD First;
    MyNumber = pThArg->iTh;
    First = pThArg->LowRec;
    RecsInGrp = pThArg->HighRec - First;
    qsort(First, RecsInGrp, RECSIZE, KeyCompare);
    while ((MyNumber % GrpSize) == 0 && RecsInGrp < nRec) {
        /* Объединить слиянием отсортированные массивы. */
        WaitForSingleObject(ThreadHandle[MyNumber + TwoToI], INFINITE);
        MergeArrays(First, First + RecsInGrp);
        RecsInGrp *= 2;
        GrpSize *= 2;
        TwoToI *= 2;
    }
    _endthreadex(0);
    return 0; /* Подавить вывод предупреждающих сообщений. */
}

```

```

static VOID MergeArrays(LPRECORD p1, LPRECORD p2) {
    DWORD iRec = 0, nRecs, i1 = 0, i2 = 0;
    LPRECORD pDest, p1Hold, pDestHold;
    nRecs = p2 - p1;
    pDest = pDestHold = malloc(2 * nRecs * RECSIZE);
    p1Hold = p1;
    while (i1 < nRecs && i2 < nRecs) {
        if (KeyCompare((LPCTSTR)p1, (LPCTSTR)p2) <= 0) {
            memcpy(pDest, p1, RECSIZE);
            i1++;
            p1++;
            pDest++;
        } else {
            memcpy(pDest, p2, RECSIZE);

```



```

    i2++;
    p2++;
    pDest++;
}
}
if (i1 >= nRecs) memcpy(pDest, p2, RECSIZE * (nRecs - i2));
else memcpy(pDest, p1, RECSIZE * (nRecs - i1));
memcpy(p1Hold, pDestHold, 2 * nRecs * RECSIZE);
free (pDestHold);
return;
}

```

### **Производительность**

В приложении В представлены результаты сортировки файлов большого размера, содержащих записи длиной 64 байта, для случаев использования одной, двух и четырех потоков. SMP-системы позволяют получать значительно лучшие результаты. Упомянутый принцип "разделяй и властвуй" обеспечивает нечто большее, чем просто стратегию проектирования алгоритмов; он также служит ключом к использованию потоков и SMP. Результаты для однопроцессорных систем могут быть различными в зависимости от остальных характеристик системы. В системах с ограниченным объемом памяти (то есть объема физической памяти не достаточно для того, чтобы наряду с ОС и другими активными процессами в ней уместился весь файл) использование нескольких потоков увеличивает время сортировки, поскольку потоки состязаются между собой в захвате доступной физической памяти. С другой стороны, если памяти имеется достаточно, то многопоточный вариант может привести к повышению производительности и в случае однопроцессорных систем. Кроме того, как следует из приложения В, получаемые результаты существенно зависят от начального распределения данных.

**Задание 2.** Воспроизведите текст программы. Выполните его.

**Задание 3.** Подготовить отчет о выполненной работе

### **Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

## **Практическое занятие № 13, 14**

**Тема раздела:** Взаимодействие между процессами

**Тема практической работы:** Создание и открытие почтового ящика

**Тема практической работы:** Создание, подключение и именование каналов и почтовых ящиков

**Цель:** ознакомиться с работой с почтовыми ящиками.

**Планируемые результаты:**

**уметь:** работать с почтовыми ящиками

**знать:** что такое почтовый ящик.

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы; ; компьютерная операционная система Windows.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**

**Задание 1.** Ознакомиться с теоретическими положениями

Книга: Системное программирование в среде Windows

#### **Почтовые ящики**

Как и именованные каналы, почтовые ящики (mailslots) Windows снабжаются именами, которые могут быть использованы для обеспечения взаимодействия между независимыми каналами. Почтовые ящики представляют собой широковещательный механизм, основанный на дейтаграммах (описаны в главе 12), и ведут себя иначе по сравнению с именованными каналами, что делает их весьма полезными в ряде ограниченных ситуаций, которые, тем не менее, представляют большой интерес. Из наиболее важных свойств почтовых ящиков можно отметить следующие:

- Почтовые ящики являются однонаправленными.
- С одним почтовым ящиком могут быть связаны несколько записывающих программ (writers) и несколько считывающих программ (readers), но они часто связаны между собой отношениями "один ко многим" в той или иной форме.
- Записывающей программе (клиенту) не известно достоверно, все ли, только некоторые или какая-то одна из программ считывания (сервер) получили сообщение.
- Почтовые ящики могут находиться в любом месте сети.
- Размер сообщений ограничен.

Использование почтовых ящиков требует выполнения следующих операций:

- Каждый сервер создает дескриптор почтового ящика с помощью функции CreateMailSlot.
- После этого сервер ожидает получения почтового сообщения, используя функцию ReadFile.
- Клиент, обладающий только правами записи, должен открыть почтовый ящик, вызвав функцию CreateFile, и записать сообщения, используя функцию WriteFile. В случае отсутствия ожидающих программ считывания попытка открытия почтового ящика завершится ошибкой (наподобие "имя не найдено").

Сообщение клиента может быть прочитано всеми серверами; все серверы получают одно и то же сообщение.

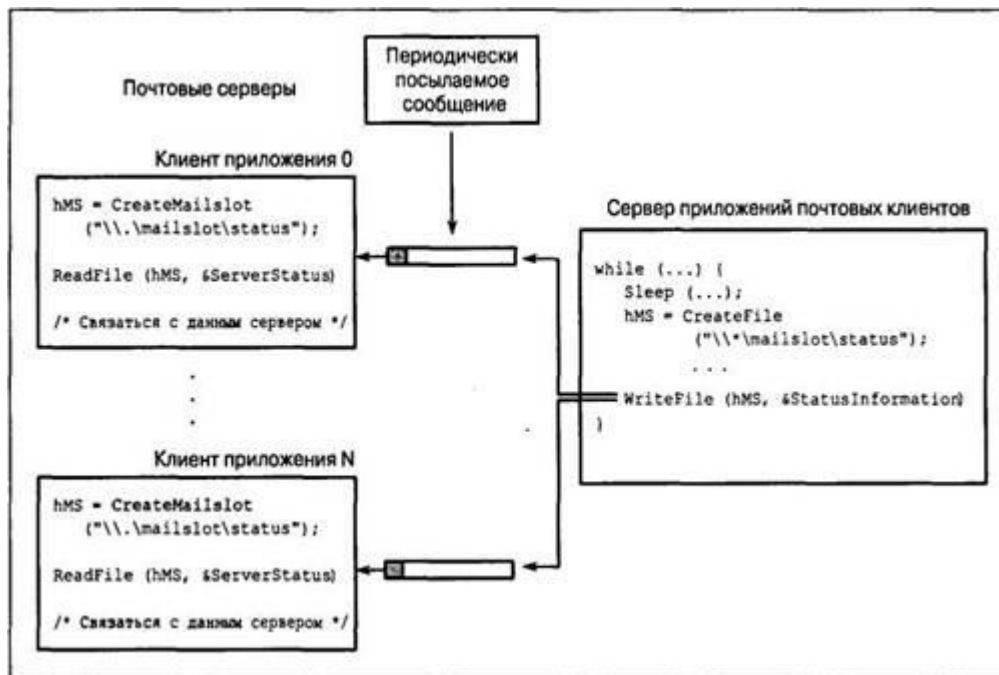
Существует еще одна возможность. В вызове функции CreateFile клиент может указать имя почтового ящика в следующем виде:

\\\*\mailslot\mailslotname

При этом символ звездочки (\*) действует в качестве группового символа (wildcard), и клиент может обнаружить любой сервер в пределах имени домена — группы систем, объединенных общим именем, которое назначается администратором сети.

### Создание и открытие почтового ящика

Для создания почтового ящика и получения дескриптора, который можно будет использовать в операциях ReadFile, почтовые серверы (программы считывания) вызывают функцию CreateMailslot. На одном компьютере может находиться только один почтовый ящик с данным именем, но один и тот же почтовый ящик может использоваться несколькими системами в сети, что обеспечивает возможность работы с ним нескольких программ считывания.



**Рис. 11.3.** Использование клиентами почтового ящика для обнаружения сервера

*HANDLE CreateMailslot(LPCTSTR lpName, DWORD cbMaxMsg, DWORD dwReadTimeout, LPSECURITY\_ATTRIBUTES lpsa)*

Параметры

lpName — указатель на строку с именем почтового ящика, которое должно иметь следующий вид:

\\.\mailslot\[путь]имя

Имя должно быть уникальным. Точка (.) указывает на то, что почтовый ящик создается на локальном компьютере.

cbMaxMsg — максимальный размер сообщения (в байтах), которые может записывать клиент. Значению 0 соответствует отсутствие ограничений.

dwReadTimeOut — длительность интервала ожидания (в миллисекундах) для операции чтения. Значению 0 соответствует немедленный возврат, а значению MAILSLT\_WAIT\_FOREVER — неопределенный период ожидания (который может длиться сколь угодно долго).

Во время открытия почтового ящика с помощью функции CreateFile клиент (записывающая программа) может указывать его имя в следующем виде:

- \\.\mailslot\[путь]имя — определяет локальный почтовый ящик. Примечание. В Windows 95 длина имени ограничена 11 символами.
- \\имя\_компьютера\mailslot\[путь]имя — определяет почтовый ящик, расположенный на компьютере с заданным именем.
- \\имя\_домена\mailslot\[путь]имя — определяет все почтовые ящики с данным именем, расположенные на компьютерах, принадлежащих данному домену. В этом случае максимальный размер сообщения составляет 424 байта.
- \\\*\mailslot\[путь]имя — определяет все почтовые ящики с данным именем, расположенные на компьютерах, принадлежащих главному домену системы. В этом случае максимальный размер сообщения составляет 424 байта.

Наконец, клиент должен указывать флаг FILE\_SHARE\_READ. Функции GetMailslotInfo и SetMailslotInfo похожи на свои аналоги, работающие с именованными каналами.

*Средства, сопоставимые с почтовыми ящиками, в UNIX отсутствуют. Однако для этой цели могут быть использованы широковещательные (broadcast) или групповые (multicast) дейтаграммы протокола TCP/IP.*

### **Использование почтовых ящиков**

Рассмотренный перед этим клиент-серверный процессор командной строки предполагает несколько возможных способов его использования. Рассмотрим один из сценариев, в котором решается задача обнаружения сервера в только что упомянутой клиент-серверной системе (программы 11.2 и 11.3).

Сервер приложения (application server), действуя в качестве почтового клиента (mailslot client), периодически осуществляет широковещательную рассылку своего имени и имени именованного канала. Любой клиент приложения (application client), которому требуется найти сервер, может получить это имя, действуя в качестве сервера почтовых ящиков (mailslot server). Аналогичным образом сервер командной строки может периодически осуществлять широковещательную рассылку своего состояния, включая информацию о коэффициенте использования, клиентам. Это соответствует ситуации, в которой имеется одна записывающая программа (почтовый клиент) и несколько считывающих программ (почтовых серверов). Если бы почтовых клиентов (то есть серверов приложения) было несколько, то ситуация описывалась бы отношением типа "многие ко многим".

Возможен и другой вариант, когда одна считывающая программа получает сообщения от многочисленных записывающих программ, которые, например, предоставляют информацию о своем состоянии. Этот вариант, соответствующий, например, электронной доске объявлений, оправдывает использование термина почтовый ящик. Оба описанных варианта использования — широковещательная рассылка имени и информации о состоянии — могут быть объединены, чтобы клиент мог выбирать наиболее подходящий сервер.

Обмен ролями терминов клиент и сервер в данном контексте может несколько сбивать с толку, однако заметьте, что сервер именованного канала и почтовый сервер выполняют вызовы функций CreateNamedPipe (или CreateMailSlot), тогда как клиент (именованного канала или почтового ящика) создает соединение, используя функцию CreateFile. Кроме того, в обоих случаях первый вызов функции WriteFile выполняется клиентом, а первый вызов функции ReadFile выполняется сервером.

Использование почтовых ящиков в соответствии с первым из описанных возможных вариантов иллюстрируется на рис. 11.3.

#### **создание, подключение и именование каналов и почтовых ящиков**

В табл. 11.1 сведены все допустимые формы имен каналов, которые могут использоваться клиентами и серверами приложения. Здесь же перечислены все функции, которые следует использовать для создания именованных каналов и соединения с ними.

Аналогичная информация для почтовых ящиков приведена в табл. 11.2. Вспомните, что почтовый клиент (или сервер) не обязательно должен выполняться тем же процессом или даже на той же системе, что и клиент (или сервер) приложения.

---

Таблица 11.1. Именованные каналы: создание, подключение и именование

Дескриптор именованного канала	Клиент приложения		Сервер приложения
	CreateFile	CallNamedPipe	CreateNamedPipe
	TransactNamedPipe		ipe

или соединение

<b>Имя канала</b>	\\\\.\\имя канала (канал является локальным) \\\имя (канал системы\\имя канала (канал является локальным удаленным)
-------------------	---------------------------------------------------------------------------------------------------------------------

Таблица 11.2. Почтовые ящики: создание, подключение и именование

	Почтовый клиент	Почтовый сервер
<b>Дескриптор</b>	CreateFile	CreateMailslot
<b>почтового ящика</b>		
<b>Имя</b>	\\\\.\\имя почтового ящика (почтовый ящик является локальным) \\\имя почтового ящика (почтовый ящик располагается на указанной удаленной системе) \\\*\\имя почтового ящика (все почтовые ящики, имеющие одно и то же указанное имя)	\\\\.\\имя почтового ящика (почтовый ящик является локальным)

#### Пример: сервер, обнаруживаемый клиентами

Программа 11.4 представляет функцию потока, которую сервер командной строки (программа 11.3), выступающий в роли почтового клиента, использует для широковещательной рассылки имени своего канала ожидающим клиентам. Возможно существование нескольких серверов с различными характеристиками и именами каналов, и клиенты получают их имена, используя почтовый ящик с известным именем. Эта функция запускается как поток программой 11.3.

#### Примечание

*На практике многие клиент-серверные системы инвертируют используемую здесь логику поиска. Суть альтернативного варианта заключается в том, что клиент приложения действует и как почтовый клиент, осуществляя широковещательную рассылку сообщений, требующих, чтобы сервер ответил с использованием указанного именованного канала (имя канала определяется клиентом и включается в сообщение). Затем сервер приложения, действующий в качестве почтового сервера, считывает запрос и создает соединение с использованием указанного именованного канала.*

Программа 11.4. SrvrBcst: функция потока почтового клиента

```
static DWORD WINAPI ServerBroadcast(LPVOID pNull) {
    MS_MESSAGE MsNotify;
    DWORD nXfer;
    HANDLE hMsFile;
```

```

/*Открыть почтовый ящик для записывающей программы почтового
"клиента"*/
while (!ShutDown) { /* Цикл выполняется до тех пор, пока имеются
серверные потоки. */
    /* Ждать, пока другой клиент не откроет почтовый ящик. */
    Sleep(CS_TIMEOUT);
    hMsFile = CreateFile(MS_CLTNAME, GENERIC_WRITE,
FILE_SHARE_READ, NULL, OPEN_EXISTING, FILE_ATTRIBUTE
NORMAL, NULL);
    if (hMsFile == INVALID_HANDLE_VALUE) continue;
    /* Отправить сообщение в почтовый ящик. */
    MsNotify.msStatus = 0;
    MsNotify.msUtilization = 0;
    _tcsncpy(MsNotify.msName, SERVER_PIPE);
    if (WriteFile(hMsFile, &MsNotify, MSM_SIZE, &nXfer, NULL))
ReportError(_T("Ошибка записи почтового сервера."), 13, TRUE);
    CloseHandle(hMsFile);
}
_tprintf(_T("Закрытие контролирующего потока.\n"));
_endthreadex(0);
return 0;
}

```

В программе 11.5 представлена функция, которая вызывается клиентом (см. программу 11.2) для обнаружения сервера.

Программа 11.5. LocSrvr: почтовый сервер

/\* Глава 11. LocSrver.c \*/

/\* Найти сервер путем считывания информации из почтового ящика, используемого для широковещательной рассылки имен серверов. \*/

---

```

#include "EvryThng.h"

```

```

#include "ClntSrvr.h" /* Определяет имя почтового ящика. */

```

---

```

BOOL LocateServer(LPTSTR pPipeName) {
    HANDLE MsFile;
    MS_MESSAGE ServerMsg;
    BOOL Found = FALSE;
    DWORD cbRead;
    MsFile = CreateMailslot(MS_SRVNAME, 0, CS_TIMEOUT, NULL);
    while (!Found) {
        _tprintf(_T("Поиск сервера.\n"));
        Found = ReadFile(MsFile, &ServerMsg, MSM_SIZE, &cbRead, NULL);
    }
    _tprintf(_T("Сервер найден.\n"));
    CloseHandle(MsFile);
    /* Имя канала сервера. */
}

```

```

    _tcscpy(pPipeName, ServerMsg.msName);
    return TRUE;
}

```

**Задание 2.** Воспроизведите текст программы. Выполните его.

**Задание 3.** Подготовить отчет о выполненной работе

#### **Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

### **Практическое занятие № 15, 16, 17, 18, 19, 20**

**Тема раздела:** Сетевое программирование с помощью сокетов

**Тема практической работы 15:** Подготовка и подключение клиентских запросов соединения

**Тема практической работы 16:** Подключение клиента к серверу

**Тема практической работы 17:** Прием и отправка сообщений сокетом

**Тема практической работы 18:** Клиент на основе сокета

**Тема практической работы 19:** Сервер на основе сокетов

**Тема практической работы 20:** Многопоточная DLL для обмена сообщениями через сокет

**Цель:** ознакомиться с сетевым программированием с помощью сокетов.

**Планируемые результаты:**

**уметь:** использовать возможности сокетов.

**знать:** понятие «сокет».

**Порядок выполнения задания, методические указания:**

- ознакомиться с теоретическими положениями по данной теме;
- выполнить задания практической работы;
- сформулировать вывод.

**Содержание отчета:** отчет по практической работе должен содержать: основные определения, рассуждения по выполнению заданий, необходимые изображения, вывод по работе.

**Материально-техническое и комплексно-методическое обеспечение:** методические рекомендации к выполнению работы;; компьютерная операционная система Windows XP.

**Время выполнения:** 90 минут.

**Этапы выполнения работы:**



## **Задание 1.** Ознакомиться с теоретическими положениями

Книга: Системное программирование в среде Windows

### **Сокеты Windows**

Winsock API разрабатывался как расширение Berkley Sockets API для среды Windows и поэтому поддерживается всеми системами Windows. К преимуществам Winsock можно отнести следующее:

- Перенос уже имеющегося кода, написанного для Berkeley Sockets API, осуществляется непосредственно.
- Системы Windows легко встраиваются в сети, использующие как версию IPv4 протокола TCP/IP, так и постепенно распространяющуюся версию IPv6. Помимо всего остального, версия IPv6 допускает использование более длинных IP-адресов, преодолевая существующий 4-байтовый адресный барьер версии IPv4.
- Сокеты могут использоваться совместно с перекрывающимся вводом/выводом Windows (глава 14), что, помимо всего прочего, обеспечивает возможность масштабирования серверов при увеличении количества активных клиентов.
- Сокеты можно рассматривать как дескрипторы (типа HANDLE) файлов при использовании функций ReadFile и WriteFile и, с некоторыми ограничениями, при использовании других функций, точно так же, как в качестве дескрипторов файлов сокеты применяются в UNIX. Эта возможность оказывается удобной в тех случаях, когда требуется использование асинхронного ввода/вывода и портов завершения ввода/вывода.
- Существуют также дополнительные, непереносимые расширения.

### **Инициализация Winsock**

Winsock API поддерживается библиотекой DLL (WS2\_32.DLL), для получения доступа к которой следует подключить к программе библиотеку WS\_232.LIB. Эту DLL следует инициализировать с помощью нестандартной, специфической для Winsock функции WSAStartup, которая должна быть первой из функций Winsock, вызываемых программой. Когда необходимость в использовании функциональных возможностей Winsock отпадает, следует вызывать функцию WSACleanup. Примечание. Префикс WSA означает "Windows Sockets asynchronous ..." ("Асинхронный Windows Sockets ..."). Средства асинхронного режима Winsock нами здесь не используются, поскольку при возникновении необходимости в выполнении асинхронных операций мы можем и будем использовать потоки.

Хотя функции WSAStartup и WSACleanup необходимо вызывать в обязательном порядке, вполне возможно, что они будут единственными нестандартными функциями, с которыми вам придется иметь дело. Распространенной практикой является применение директив препроцессора #ifdef для проверки значения символической константы \_WIN32 (обычно определяется Visual C++ на стадии компиляции), в результате чего функции WSA будут вызываться только тогда, когда вы работаете в Windows).

Разумеется, такой подход предполагает, что остальная часть кода не зависит от платформы.

```
int WSAStartup(WORD wVersionRequired, LPWSADATA lpWSADATA);
```

Параметры

*wVersionRequired* — указывает старший номер версии библиотеки DLL, который вам требуется и который вы можете использовать. Как правило, версии 1.1 вполне достаточно для того, чтобы обеспечить любое взаимодействие с другими системами, в котором у вас может возникнуть необходимость. Тем не менее, во всех системах Windows, включая Windows 9x, доступна версия Winsock 2.0, которая и используется в приведенных ниже примерах. Версия 1.1 считается устаревшей и постепенно выходит из употребления.

Функция возвращает ненулевое значение, если запрошенная вами версия данной DLL не поддерживается.

Младший байт параметра *wVersionRequired* указывает основной номер версии, а старший байт — дополнительный. Обычно используют макрос *MAKEWORD*; таким образом, выражение *MAKEWORD(2,0)* представляет версию 2.0.

*lpWSADATA* — указатель на структуру *WSADATA*, которая возвращает информацию о конфигурации DLL, включая старший доступный номер версии. О том, как интерпретировать ее содержимое, вы можете прочитать в материалах оперативной справки Visual Studio.

Чтобы получить более подробную информацию об ошибках, можно воспользоваться функцией *WSAGetLastError*, но для этой цели подходит также функция *GetLastError*, а также функция *ReportError*, разработанная в главе 2.

По окончании работы программы, а также в тех случаях, когда необходимости в использовании сокетов больше нет, следует вызывать функцию *WSACleanup*, чтобы библиотека *WS\_32.DLL*, обслуживающая сокет, могла освободить ресурсы, распределенные для этого процесса.

### **Создание сокета**

Инициализировав Winsock DLL, вы можете использовать стандартные (Berkeley Sockets) функции для создания сокетов и соединений, обеспечивающих взаимодействие серверов с клиентами или взаимодействие равноправных узлов сети между собой.

Используемый в Winsock тип данных *SOCKET* аналогичен типу данных *HANDLE* в Windows, и его даже можно применять совместно с функцией *ReadFile* и другими функциями Windows, требующими использования дескрипторов типа *HANDLE*. Для создания (или открытия) сокета служит функция *socket*.

```
SOCKET socket(int af, int type, int protocol);
```

Параметры

Тип данных *SOCKET* фактически определяется как тип данных *int*, потому код UNIX остается переносимым, не требуя привлечения типов данных Windows.

af — обозначает семейство адресов, или протокол; для указания протокола IP (компонент протокола TCP/IP, отвечающий за протокол Internet) следует использовать значение PF\_INET (или AF\_INET, которое имеет то же самое числовое значение, но обычно используется при вызове функции bind).

type — указывает тип взаимодействия: ориентированное на установку соединения (connection-oriented communication), или потоковое (SOCK\_STREAM), и дейтаграммное (datagram communication) (SOCK\_DGRAM), что в определенной степени сопоставимо соответственно с именованными каналами и почтовыми ящиками.

protocol — является излишним, если параметр af установлен равным AF\_INET; используйте значение 0.

В случае неудачного завершения функция socket возвращает значение INVALID\_SOCKET.

Winsock можно использовать совместно с протоколами, отличными от TCP/IP, указывая различные значения параметра protocol; мы же будем использовать только протокол TCP/IP.

Как и в случае всех остальных стандартных функций, имя функции socket не должно содержать прописных букв. Это является отходом от соглашений, принятых в Windows, и продиктовано необходимостью соблюдения промышленных стандартов.

Winsock API разрабатывался как расширение Berkley Sockets API для среды Windows и поэтому поддерживается всеми системами Windows. К преимуществам Winsock можно отнести следующее:

- Перенос уже имеющегося кода, написанного для Berkeley Sockets API, осуществляется непосредственно.
- Системы Windows легко встраиваются в сети, использующие как версию IPv4 протокола TCP/IP, так и постепенно распространяющуюся версию IPv6. Помимо всего остального, версия IPv6 допускает использование более длинных IP-адресов, преодолевая существующий 4-байтовый адресный барьер версии IPv4.
- Сокеты могут использоваться совместно с перекрывающимся вводом/выводом Windows (глава 14), что, помимо всего прочего, обеспечивает возможность масштабирования серверов при увеличении количества активных клиентов.
- Сокеты можно рассматривать как дескрипторы (типа HANDLE) файлов при использовании функций ReadFile и WriteFile и, с некоторыми ограничениями, при использовании других функций, точно так же, как в качестве дескрипторов файлов сокеты применяются в UNIX. Эта возможность оказывается удобной в тех случаях, когда требуется использование асинхронного ввода/вывода и портов завершения ввода/вывода.
- Существуют также дополнительные, непереносимые расширения.

#### **Создание сокета**

Инициализировав Winsock DLL, вы можете использовать стандартные (Berkeley Sockets) функции для создания сокетов и соединений, обеспечивающих взаимодействие серверов с клиентами или взаимодействие равноправных узлов сети между собой.

Используемый в Winsock тип данных SOCKET аналогичен типу данных HANDLE в Windows, и его даже можно применять совместно с функцией ReadFile и другими функциями Windows, требующими использования дескрипторов типа HANDLE. Для создания (или открытия) сокета служит функция socket.

*SOCKET socket(int af, int type, int protocol);*

Параметры

Тип данных SOCKET фактически определяется как тип данных int, потому код UNIX остается переносимым, не требуя привлечения типов данных Windows.

af — обозначает семейство адресов, или протокол; для указания протокола IP (компонент протокола TCP/IP, отвечающий за протокол Internet) следует использовать значение PF\_INET (или AF\_INET, которое имеет то же самое числовое значение, но обычно используется при вызове функции bind).

type — указывает тип взаимодействия: ориентированное на установку соединения (connection-oriented communication), или потоковое (SOCK\_STREAM), и дейтаграммное (datagram communication) (SOCK\_DGRAM), что в определенной степени сопоставимо соответственно с именованными каналами и почтовыми ящиками.

protocol — является излишним, если параметр af установлен равным AF\_INET; используйте значение 0.

В случае неудачного завершения функция socket возвращает значение INVALID\_SOCKET.

Winsock можно использовать совместно с протоколами, отличными от TCP/IP, указывая различные значения параметра protocol; мы же будем использовать только протокол TCP/IP.

Как и в случае всех остальных стандартных функций, имя функции socket не должно содержать прописных букв. Это является отходом от соглашений, принятых в Windows, и продиктовано необходимостью соблюдения промышленных стандартов.

### **Связывание сокета**

Следующий шаг заключается в привязке сокета к его адресу и конечной точке (endpoint) (направление канала связи от приложения к службе). Вызов socket, за которым следует вызов bind, аналогичен созданию именованного канала. Однако не существует имен, используя которые можно было бы различать сокеты данного компьютера. Вместо этого в качестве конечной точки службы используется номер порта (port number). Любой заданный сервер может иметь несколько конечных точек. Прототип функции bind приводится ниже.

*int bind(SOCKET s, const struct sockaddr \*saddr, int namelen);*

## Параметры

*s* — несвязанный сокет, возвращенный функцией `socket`.

*saddr* — заполняется перед вызовом и задает протокол и специфическую для протокола информацию, как описано ниже. Кроме всего прочего, в этой структуре содержится номер порта.

*namelen* — присвойте значение `sizeof (sockaddr)`.

В случае успешного выполнения функция возвращает значение 0, иначе `SOCKET_ERROR`. Структура `sockaddr` определяется следующим образом:

```
struct sockaddr {  
    u_short sa_family;  
    char sa_data[14] ;  
};  
typedef struct sockaddr SOCKADDR, *PSOCKADDR;
```

Первый член этой структуры, *sa\_family*, обозначает протокол. Второй член, *sa\_data*, зависит от протокола. Internet-версией структуры *sa\_data* является структура `sockaddr_in`:

```
struct sockaddr_in {  
    short sin_family; /* AF_INET */  
    u_short sin_port;  
    struct in_addr sin_addr; /* 4-байтовый IP-адрес */  
    char sin_zero[8];  
};  
typedef struct sockaddr_in SOCKADDR_IN, *PSOCKADDR_IN;
```

Обратите внимание на использование типа данных `short integer` для номера порта. Кроме того, номер порта и иная информация должны храниться с соблюдением подходящего порядка следования байтов, при котором старший байт помещается в крайней позиции справа (`big-endian`), чтобы обеспечивалась двоичная совместимость с другими системами. В структуре `sin_addr` содержится подструктура `s_addr`, заполняемая уже знакомым нам 4-байтовым IP-адресом, например 127.0.0.1, указывающим систему, чей запрос на образование соединения должен быть принят. Обычно удовлетворяются запросы любых систем, в связи с чем следует использовать значение `INADDR_ANY`, хотя этот символический параметр должен быть преобразован к корректному формату, как показано в приведенном ниже фрагменте кода.

Для преобразования текстовой строки с IP-адресом к требуемому формату можно использовать функцию `inet_addr`, поэтому член `sin_addr.s_addr` переменной `sockaddr_in` инициализируется следующим образом:

```
sa.sin_addr.s_addr = inet_addr("192.13.12.1");
```

О связанном сокете, для которого определены протокол, номер порта и IP-адрес, иногда говорят как об именованном сокете (`named socket`).

## Перевод связанного сокета в состояние прослушивания

Функция `listen` делает сервер доступным для образования соединения с клиентом. Аналогичной функции для именованных каналов не существует.

*int listen(SOCKET s, int nQueueSize);*

Параметр `nQueueSize` указывает число запросов на соединение, которые вы намерены помещать в очередь сокета. В версии Winsock 2.0 значение этого параметра не имеет ограничения сверху, но в версии 1.1 оно ограничено предельным значением `SOMAXCON` (равным 5).

### **Прием клиентских запросов соединения**

Наконец, сервер может ожидать соединения с клиентом, используя функцию `accept`, возвращающую новый подключенный сокет, который будет использоваться в операциях ввода/вывода. Заметьте, что исходный сокет, который теперь находится в состоянии прослушивания (`listening state`), используется исключительно в качестве параметра функции `accept`, а не для непосредственного участия в операциях ввода/вывода.

Функция `accept` блокируется до тех пор, пока от клиента не поступит запрос соединения, после чего она возвращает новый сокет ввода/вывода. Хотя рассмотрение этого и выходит за рамки данной книги, возможно создание неблокирующихся сокетов, а в сервере (программа 12.2) для приема запроса используется отдельный поток, что позволяет создавать также неблокирующиеся серверы.

*SOCKET accept(SOCKET s, LP SOCKADDR lpAddr, LP INT lpAddrLen);*

Параметры

`s` — прослушивающий сокет. Чтобы перевести сокет в состояние прослушивания, необходимо предварительно вызвать функции `socket`, `bind` и `listen`.

`lpAddr` — указатель на структуру `sockaddr_in`, предоставляющую адрес клиентской системы.

`lpAddrLen` — указатель на переменную, которая будет содержать размер возвращенной структуры `sockaddr_in`. Перед вызовом функции `accept` эта переменная должна быть инициализирована значением `sizeof(struct sockaddr_in)`.

### **Отключение и закрытие сокетов**

Для отключения сокетов применяется функция `shutdown(s, how)`. Аргумент `how` может принимать одно из двух значений: 1, указывающее на то, что соединение может быть разорвано только для отправки сообщений, и 2, соответствующее разрыву соединения как для отправки, так и для приема сообщений. Функция `shutdown` не освобождает ресурсы, связанные с сокетом, но гарантирует завершение отправки и приема всех данных до закрытия сокета. Тем не менее, после вызова функции `shutdown` приложение уже не должно использовать этот сокет.

Когда работа с сокетом закончена, его следует закрыть, вызвав функцию `closesocket(SOCKET s)`. Сначала сервер закрывает сокет, созданный функцией `accept`, а не прослушивающий сокет, созданный с помощью функции `socket`. Сервер должен закрывать прослушивающий сокет только тогда, когда завершает работу или прекращает принимать клиентские

запросы соединения. Даже если вы работаете с сокетом как с дескриптором типа HANDLE и используете функции ReadFile и WriteFile, уничтожить сокет одним только вызовом функции CloseHandle вам не удастся; для этого следует использовать функцию closesocket.

### **Пример: подготовка и получение клиентских запросов соединения**

Ниже приводится фрагмент кода, показывающий, как создать сокет и организовать прием клиентских запросов соединения.

В этом примере используются две стандартные функции: htons ("host to network short" — "ближняя связь") и htonl ("host to network long" — "дальняя связь"), которые преобразуют целые числа к форме с обратным порядком байтов, требуемой протоколом IP.

Номером порта сервера может быть любое число из диапазона, допустимого для целых чисел типа short integer, но для определенных пользователем служб обычно используются числа в диапазоне 1025—5000. Порты с меньшими номерами зарезервированы для таких известных служб, как telnet или ftp, в то время как порты с большими номерами предполагаются для использования других стандартных служб.

```
struct sockaddr_in SrvSAddr; /* Адресная структура сервера. */
struct sockaddr_in ConnectAddr;
SOCKET SrvSock, sockio;
...
SrvSock = socket(AF_INET, SOCK_STREAM, 0);
SrvSAddr.sin_family = AF_INET;
SrvSAddr.sin_addr.s_addr = htonl(INADDR_ANY);
SrvSAddr.sin_port = htons(SERVER_PORT);
bind(SrvSock, (struct sockaddr *)&SrvSAddr, sizeof SrvSAddr);
listen(SrvSock, 5);
AddrLen = sizeof(ConnectAddr);
sockio = accept(SrvSock, (struct sockaddr *)&ConnectAddr, &AddrLen);
... Получение запросов и отправка ответов ...
shutdown(sockio);
closesocket(sockio);
```

### **Серверные функции сокета**

В нижеследующем обсуждении под сервером будет пониматься процесс, который принимает запросы на образование соединения через заданный порт. Несмотря на то что сокеты, подобно именованным каналам, могут использоваться для создания соединений между равноправными узлами сети, введение указанного различия между узлами является весьма удобным и отражает различия в способах, используемых обеими системами для соединения друг с другом.

Если не оговорено иное, типом сокетов в наших примерах всегда будет SOCK\_STREAM. Сокеты типа SOCK\_DGRAM рассматривается далее в этой главе.

### **Связывание сокета**

Следующий шаг заключается в привязке сокета к его адресу и конечной точке (endpoint) (направление канала связи от приложения к службе). Вызов `socket`, за которым следует вызов `bind`, аналогичен созданию именованного канала. Однако не существует имен, используя которые можно было бы различать сокеты данного компьютера. Вместо этого в качестве конечной точки службы используется номер порта (port number). Любой заданный сервер может иметь несколько конечных точек. Прототип функции `bind` приводится ниже.

```
int bind(SOCKET s, const struct sockaddr *saddr, int namelen);
```

Параметры

`s` — несвязанный сокет, возвращенный функцией `socket`.

`saddr` — заполняется перед вызовом и задает протокол и специфическую для протокола информацию, как описано ниже. Кроме всего прочего, в этой структуре содержится номер порта.

`namelen` — присвойте значение `sizeof (sockaddr)`.

В случае успешного выполнения функция возвращает значение 0, иначе `SOCKET_ERROR`. Структура `sockaddr` определяется следующим образом:

```
struct sockaddr {  
    u_short sa_family;  
    char sa_data[14];  
};  
typedef struct sockaddr SOCKADDR, *PSOCKADDR;
```

Первый член этой структуры, `sa_family`, обозначает протокол. Второй член, `sa_data`, зависит от протокола. Internet-версией структуры `sa_data` является структура `sockaddr_in`:

```
struct sockaddr_in {  
    short sin_family; /* AF_INET */  
    u_short sin_port;  
    struct in_addr sin_addr; /* 4-байтовый IP-адрес */  
    char sin_zero[8];  
};  
typedef struct sockaddr_in SOCKADDR_IN, *PSOCKADDR_IN;
```

Обратите внимание на использование типа данных `short integer` для номера порта. Кроме того, номер порта и иная информация должны храниться с соблюдением подходящего порядка следования байтов, при котором старший байт помещается в крайней позиции справа (big-endian), чтобы обеспечивалась двоичная совместимость с другими системами. В структуре `sin_addr` содержится подструктура `s_addr`, заполняемая уже знакомым нам 4-байтовым IP-адресом, например 127.0.0.1, указывающим систему, чей запрос на образование соединения должен быть принят. Обычно удовлетворяются запросы любых систем, в связи с чем следует использовать значение `INADDR_ANY`, хотя этот символический параметр должен быть преобразован к корректному формату, как показано в приведенном ниже фрагменте кода.



Для преобразования текстовой строки с IP-адресом к требуемому формату можно использовать функцию `inet_addr`, поэтому член `sin_addr.s_addr` переменной `sockaddr_in` инициализируется следующим образом:

```
sa.sin_addr.s_addr = inet_addr("192.13.12.1");
```

О связанном сокете, для которого определены протокол, номер порта и IP-адрес, иногда говорят как об именованном сокете (`named socket`).

### **Перевод связанного сокета в состояние прослушивания**

Функция `listen` делает сервер доступным для образования соединения с клиентом. Аналогичной функции для именованных каналов не существует.

```
int listen(SOCKET s, int nQueueSize);
```

Параметр `nQueueSize` указывает число запросов на соединение, которые вы намерены помещать в очередь сокета. В версии Winsock 2.0 значение этого параметра не имеет ограничения сверху, но в версии 1.1 оно ограничено предельным значением `SOMAXCON` (равным 5).

### **Прием клиентских запросов соединения**

Наконец, сервер может ожидать соединения с клиентом, используя функцию `accept`, возвращающую новый подключенный сокет, который будет использоваться в операциях ввода/вывода. Заметьте, что исходный сокет, который теперь находится в состоянии прослушивания (`listening state`), используется исключительно в качестве параметра функции `accept`, а не для непосредственного участия в операциях ввода/вывода.

Функция `accept` блокируется до тех пор, пока от клиента не поступит запрос соединения, после чего она возвращает новый сокет ввода/вывода. Хотя рассмотрение этого и выходит за рамки данной книги, возможно создание неблокирующихся сокетов, а в сервере (программа 12.2) для приема запроса используется отдельный поток, что позволяет создавать также неблокирующиеся серверы.

```
SOCKET accept(SOCKET s, LPSOCKADDR lpAddr, LPINT lpAddrLen);
```

Параметры

`s` — прослушивающий сокет. Чтобы перевести сокет в состояние прослушивания, необходимо предварительно вызвать функции `socket`, `bind` и `listen`.

`lpAddr` — указатель на структуру `sockaddr_in`, предоставляющую адрес клиентской системы.

`lpAddrLen` — указатель на переменную, которая будет содержать размер возвращенной структуры `sockaddr_in`. Перед вызовом функции `accept` эта переменная должна быть инициализирована значением `sizeof(struct sockaddr_in)`.

### **Отключение и закрытие сокетов**

Для отключения сокетов применяется функция `shutdown(s, how)`. Аргумент `how` может принимать одно из двух значений: 1, указывающее на то, что соединение может быть разорвано только для отправки сообщений, и 2, соответствующее разрыву соединения как для отправки, так и для приема сообщений. Функция `shutdown` не освобождает ресурсы, связанные с

сокетом, но гарантирует завершение отправки и приема всех данных до закрытия сокета. Тем не менее, после вызова функции `shutdown` приложение уже не должно использовать этот сокет.

Когда работа с сокетом закончена, его следует закрыть, вызвав функцию `closesocket(SOCKET s)`. Сначала сервер закрывает сокет, созданный функцией `accept`, а не прослушивающий сокет, созданный с помощью функции `socket`. Сервер должен закрывать прослушивающий сокет только тогда, когда завершает работу или прекращает принимать клиентские запросы соединения. Даже если вы работаете с сокетом как с дескриптором типа `HANDLE` и используете функции `ReadFile` и `WriteFile`, уничтожить сокет одним только вызовом функции `CloseHandle` вам не удастся; для этого следует использовать функцию `closesocket`.

### **Пример: подготовка и получение клиентских запросов соединения**

Ниже приводится фрагмент кода, показывающий, как создать сокет и организовать прием клиентских запросов соединения.

В этом примере используются две стандартные функции: `htons` ("host to network short" — "ближняя связь") и `htonl` ("host to network long" — "дальняя связь"), которые преобразуют целые числа к форме с обратным порядком байтов, требуемой протоколом IP.

Номером порта сервера может быть любое число из диапазона, допустимого для целых чисел типа `short integer`, но для определенных пользователем служб обычно используются числа в диапазоне 1025—5000. Порты с меньшими номерами зарезервированы для таких известных служб, как `telnet` или `ftp`, в то время как порты с большими номерами предполагаются для использования других стандартных служб.

```
struct sockaddr_in SrvSAddr; /* Адресная структура сервера. */
struct sockaddr_in ConnectAddr;
SOCKET SrvSock, sockio;

...
SrvSock = socket(AF_INET, SOCK_STREAM, 0);
SrvSAddr.sin_family = AF_INET;
SrvSAddr.sin_addr.s_addr = htonl(INADDR_ANY);
SrvSAddr.sin_port = htons(SERVER_PORT);
bind(SrvSock, (struct sockaddr *)&SrvSAddr, sizeof SrvSAddr);
listen(SrvSock, 5);
AddrLen = sizeof(ConnectAddr);
sockio = accept(SrvSock, (struct sockaddr *)&ConnectAddr, &AddrLen);
... Получение запросов и отправка ответов ...
shutdown(sockio);
closesocket(sockio);
```

В нижеследующем обсуждении под сервером будет пониматься процесс, который принимает запросы на образование соединения через заданный порт. Несмотря на то что сокеты, подобно именованным каналам, могут использоваться для создания соединений между равноправными узлами сети, введение указанного различия между узлами является весьма

удобным и отражает различия в способах, используемых обеими системами для соединения друг с другом.

Если не оговорено иное, типом сокетов в наших примерах всегда будет `SOCK_STREAM`. Сокеты типа `SOCK_DGRAM` рассматривается далее в этой главе.

### **Клиентские функции сокета**

Клиентская станция, которая желает установить соединение с сервером, также должна создать сокет, вызвав функцию `socket`. Следующий шаг заключается в установке соединения сервером, а, кроме того, необходимо указать номер порта, адрес хоста и другую информацию. Имеется только одна дополнительная функция – `connect`.

### **Установка клиентского соединения с сервером**

Если имеется сервер с сокетом в режиме прослушивания, клиент может соединиться с ним при помощи функции `connect`.

```
int connect(SOCKET s, LPSOCKADDR lpName, int nNameLen);
```

Параметры

`s` — сокет, созданный с использованием функции `socket`.

`lpName` — указатель на структуру `sockaddr_in`, инициализированную значениями номера порта и IP-адреса системы с сокетом, связанным с указанным портом, который находится в состоянии прослушивания.

Инициализируйте `nNameLen` значением `sizeof (struct sockaddr_in)`.

Возвращаемое значение 0 указывает на успешное завершение функции, тогда как значение `SOCKET_ERROR` указывает на ошибку, которая, в частности, может быть обусловлена отсутствием прослушивающего сокета по указанному адресу.

Сокет `s` не обязательно должен быть связанным с портом до вызова функции `connect`, хотя это и может иметь место. При необходимости система распределяет порт и определяет протокол.

### **Пример: подключение клиента к серверу**

Показанный ниже фрагмент кода обеспечивает соединение клиента с сервером. Для этого нужны только два вызова функций, но адресная структура должна быть инициализирована до вызова функции `connect`. Проверка возможных ошибок здесь отсутствует, но в реальные программы она должна включаться. В примере предполагается, что IP-адрес (текстовая строка наподобие "192.76.33.4") задается в аргументе `argv[1]` командной строки.

```
SOCKET ClientSock;
```

```
...
```

```
ClientSock = socket(AF_INET, SOCK_STREAM, 0);
```

```
memset(&ClientSAddr, 0, sizeof(ClientSAddr));
```

```
ClientSAddr.sin_family = AF_INET;
```

```
ClientSAddr.sin_addr.s_addr = inet_addr(argv[1]);
```

```
ClientSAddr.sin_port = htons(SERVER_PORT);
```

```
ConVal = connect(ClientSock, (struct sockaddr *)&ClientSAddr,  
sizeof(ClientSAddr));
```

## Отправка и получение данных

Программы, использующие сокет, обмениваются данными с помощью функций `send` и `recv`, прототипы которых почти совпадают (перед указателем буфера функции `send` помещается модификатор `const`). Ниже представлен только прототип функции `send`.

```
int send(SOCKET s, const char * lpBuffer, int nBufferLen, int nFlags);
```

Возвращаемым значением является число фактически переданных байтов. Значение `SOCKET_ERROR` указывает на ошибку.

`nFlags` — может использоваться для обозначения степени срочности сообщений (например, экстренных сообщений), а значение `MSG_PEEK` позволяет просматривать получаемые данные без их считывания.

Самое главное, что вы должны запомнить — это то, что функции `send` и `recv` не являются атомарными (`atomic`), и поэтому нет никакой гарантии, что затребованные данные будут действительно отправлены или получены. Передача "коротких" сообщений ("short sends") встречается крайне редко, хотя и возможна, что справедливо и по отношению к приему "коротких" сообщений ("short receives"). Понятие сообщения в том смысле, который оно имело в случае именованных каналов, здесь отсутствует, и поэтому вы должны проверять возвращаемое значение и повторно отправлять или принимать данные до тех пор, пока все они не будут переданы.

С сокетами могут использоваться также функции `ReadFile` и `WriteFile`, только в этом случае при вызове функции необходимо привести сокет к типу `HANDLE`.

Клиентская станция, которая желает установить соединение с сервером, также должна создать сокет, вызвав функцию `socket`. Следующий шаг заключается в установке соединения сервером, а, кроме того, необходимо указать номер порта, адрес хоста и другую информацию. Имеется только одна дополнительная функция — `connect`.

### Установление клиентского соединения с сервером

Если имеется сервер с сокетом в режиме прослушивания, клиент может соединиться с ним при помощи функции `connect`.

```
int connect(SOCKET s, LPSOCKADDR lpName, int nNameLen);
```

Параметры

`s` — сокет, созданный с использованием функции `socket`.

`lpName` — указатель на структуру `sockaddr_in`, инициализированную значениями номера порта и IP-адреса системы с сокетом, связанным с указанным портом, который находится в состоянии прослушивания.

Инициализируйте `nNameLen` значением `sizeof (struct sockaddr_in)`.

Возвращаемое значение 0 указывает на успешное завершение функции, тогда как значение `SOCKET_ERROR` указывает на ошибку, которая, в частности, может быть обусловлена отсутствием прослушивающего сокета по указанному адресу.

Сокет `s` не обязательно должен быть связанным с портом до вызова функции `connect`, хотя это и может иметь место. При необходимости система распределяет порт и определяет протокол.

### **Пример: подключение клиента к серверу**

Показанный ниже фрагмент кода обеспечивает соединение клиента с сервером. Для этого нужны только два вызова функций, но адресная структура должна быть инициализирована до вызова функции connect. Проверка возможных ошибок здесь отсутствует, но в реальные программы она должна включаться. В примере предполагается, что IP-адрес (текстовая строка наподобие "192.76.33.4") задается в аргументе argv[1] командной строки.

```
SOCKET ClientSock;  
  
...  
ClientSock = socket(AF_INET, SOCK_STREAM, 0);  
memset(&ClientSAddr, 0, sizeof(ClientSAddr));  
ClientSAddr.sin_family = AF_INET;  
ClientSAddr.sin_addr.s_addr = inet_addr(argv[1]);  
ClientSAddr.sin_port = htons(SERVER_PORT);  
ConVal = connect(ClientSock, (struct sockaddr *)&ClientSAddr,  
sizeof(ClientSAddr));
```

### **Отправка и получение данных**

Программы, использующие сокеты, обмениваются данными с помощью функций send и recv, прототипы которых почти совпадают (перед указателем буфера функции send помещается модификатор const). Ниже представлен только прототип функции send.

```
int send(SOCKET s, const char *lpBuffer, int nBufferLen, int nFlags);
```

Возвращаемым значением является число фактически переданных байтов. Значение SOCKET\_ERROR указывает на ошибку.

nFlags — может использоваться для обозначения степени срочности сообщений (например, экстренных сообщений), а значение MSG\_PEEK позволяет просматривать получаемые данные без их считывания.

Самое главное, что вы должны запомнить — это то, что функции send и recv не являются атомарными (atomic), и поэтому нет никакой гарантии, что затребованные данные будут действительно отправлены или получены. Передача "коротких" сообщений ("short sends") встречается крайне редко, хотя и возможна, что справедливо и по отношению к приему "коротких" сообщений ("short receives"). Понятие сообщения в том смысле, который оно имело в случае именованных каналов, здесь отсутствует, и поэтому вы должны проверять возвращаемое значение и повторно отправлять или принимать данные до тех пор, пока все они не будут переданы.

С сокетами могут использоваться также функции ReadFile и WriteFile, только в этом случае при вызове функции необходимо привести сокет к типу HANDLE.

### **Пример: функция приема сообщений в случае сокета**

Часто оказывается удобным отправлять и получать сообщения в виде единых блоков. Как было показано в главе 11, каналы позволяют это сделать. Однако в случае сокетов требуется создание заголовка, содержащего размер сообщения, за которым следует само сообщение. Для приема таких

сообщений предназначена функция `ReceiveMessage`, которая будет использоваться в примерах. То же самое можно сказать и о функции `SendMessage`, предназначенной для передачи сообщений.

Обратите внимание, что сообщение принимается в виде двух частей: заголовок и содержимого. Ниже мы предполагаем, что пользовательскому типу `MESSAGE` соответствует 4-байтовый заголовок. Но даже для 4-байтового заголовка требуются повторные вызовы функции `recv`, чтобы гарантировать его полное считывание, поскольку функция `recv` не является атомарной.

***Примечание, относящееся к Win64***

*В качестве типа переменных, используемых для хранения размера сообщения, выбран тип данных фиксированной точности `LONG32`, которого будет вполне достаточно для размещения значений параметра размера, включаемого в сообщения при взаимодействии с системами, отличными от Windows, и который годится для возможной последующей перекompilляции программы для ее использования на платформе Win64 (см. главу 16).*

```
DWORD ReceiveMessage (MESSAGE *pMsg, SOCKET sd) {
    /* Сообщение состоит из 4-байтового поля размера сообщения, за
    которым следует собственно содержимое. */
    DWORD Disconnect = 0;
    LONG32 nRemainRecv, nXfer;
    LPBYTE pBuffer;
    /* Считать сообщение. */
    /* Сначала считывается заголовок, а затем содержимое. */
    nRemainRecv = 4; /* Размер поля заголовка. */
    pBuffer = (LPBYTE)pMsg; /* recv может не передать все запрошенные
    байты. */
    while (nRemainRecv > 0 && !Disconnect) {
        nXfer = recv(sd, pBuffer, nRemainRecv, 0);
        Disconnect = (nXfer == 0);
        nRemainRecv -= nXfer;
        pBuffer += nXfer;
    }
    /* Считать содержимое сообщения. */
    nRemainRecv = pMsg->RqLen;
    while (nRemainRecv > 0 && !Disconnect) {
        nXfer = recv(sd, pBuffer, nRemainRecv, 0);
        Disconnect = (nXfer == 0);
        nRemainRecv -= nXfer;
        pBuffer += nXfer;
    }
    return Disconnect;
}
```

**Пример: клиент на основе сокета**

Программа 12.1 представляет собой переработанный вариант клиентской программы clientNP (программа 11.2), которая использовалась в случае именованных каналов. Преобразование программы осуществляется самым непосредственным образом и требует лишь некоторых пояснений.

- Вместо обнаружения сервера с помощью почтовых ящиков пользователь вводит IP-адрес сервера в командной строке. Если IP-адрес не указан, используется заданный по умолчанию адрес 127.0.0.1, соответствующий локальной системе.

- Для отправки и приема сообщений применяются функции, например, ReceiveMessage, которые здесь не представлены.

- Номер порта, SERVER\_PORT, определен в заголовочном файле ClntSrvr.h.

Хотя код написан для выполнения под управлением Windows, единственная зависимость от Windows связана с использованием вызовов функций, имеющих префикс WSA.

Программа 12.1. clientSK: клиент на основе сокетов

```
/* Глава 12. clientSK.c */
/* Однопоточный клиент командной строки. */
/* ВЕРСИЯ НА ОСНОВЕ WINDOWS SOCKETS. */
/* Считывает последовательность команд для пересылки серверному
процессу*/
/* через соединение с сокетом. Дождется ответа и отображает его. */

#define _NOEXCLUSIONS /* Требуется для включения определений
сокета. */
#include "EvryThng.h"
#include "ClntSrvr.h" /* Определяет структуры записей запроса и ответа.
*/

/* Функции сообщения для обслуживания запросов и ответов. */
/* Кроме того, ReceiveResponseMessage отображает полученные
сообщения. */
static DWORD SendRequestMessage(REQUEST *, SOCKET);
static DWORD ReceiveResponseMessage(RESPONSE *, SOCKET);
struct sockaddr_in ClientSAddr; /* Адрес сокета клиента. */
int _tmain(DWORD argc, LPTSTR argv[]) {
    SOCKET ClientSock = INVALID_SOCKET;
    REQUEST Request; /* См. ClntSrvr.h. */
    RESPONSE Response; /* См. ClntSrvr.h. */
    WSADATA WStartData; /* Структура данных библиотеки сокета. */
    BOOL Quit = FALSE;
    DWORD ConVal, j;
    TCHAR PromptMsg[] = _T("\nВведите команду> ");
    TCHAR Req[MAX_RQRS_LEN];
    TCHAR QuitMsg[] = _T("$Quit");
```

```

/* Запрос: завершить работу клиента. */
TCHAR ShutMsg[] = _T("$ShutDownServer"); /* Остановить все потоки.
*/
CHAR DefaultIPAddr[] = "127.0.0.1"; /* Локальная система. */
/* Инициализировать библиотеку WSA; задана версия 2.0, но будет
работать и версия 1.1. */
WSAStartup(MAKEWORD(2, 0), &WSStartData);
/* Подключиться к серверу. */
/* Следовать стандартной процедуре вызова последовательности
функций socket/connect клиентом. */
ClientSock = socket(AF_INET, SOCK_STREAM, 0);
memset(&ClientSAddr, 0, sizeof(ClientSAddr));
ClientSAddr.sin_family = AF_INET;
if (argc >= 2) ClientSAddr.sin_addr.s_addr = inet_addr(argv [1]);
else ClientSAddr.sin_addr.s_addr = inet_addr(DefaultIPAddr);
ClientSAddr.sin_port = htons(SERVER_PORT);
/* Номер порта определен равным 1070. */
connect(ClientSock, (struct sockaddr *)&ClientSAddr,
sizeof(ClientSAddr));
/* Основной цикл для вывода приглашения на ввод команд, отправки
запроса и получения ответа. */
while (!Quit) {
    _tprintf(_T("%s"), PromptMsg);
    /* Ввод в формате обобщенных строк, но команда серверу должна
указываться в формате ASCII. */
    _fgetts(Req, MAX_RQRS_LEN-1, stdin);
    for (j = 0; j <= _tcslen(Req) Request.Record[j] = Req[j];
    /* Избавиться от символа новой строки в конце строки. */
    Request.Record[strlen(Request.Record) - 1] = '\0';
    if (strcmp(Request.Record, QuitMsg) == 0 || strcmp(Request.Record,
ShutMsg) == 0) Quit = TRUE;
    SendRequestMessage(&Request, ClientSock);
    ReceiveResponseMessage(&Response, ClientSock);
}
shutdown(ClientSock, 2); /* Запретить отсылку и прием сообщений. */
closesocket(ClientSock);
WSACleanup();
_tprintf(_T("\n****Выход из клиентской программы\n"));
return 0;
}

```

### **Пример: усовершенствованный сервер на основе сокетов**

Программа serverSK (программа 12.2) аналогична программе serverNP (программа 11.3), являясь ее видоизмененным и усовершенствованным вариантом.



- В усовершенствованном варианте программы серверные потоки создаются по требованию (on demand), а не в виде пула потоков фиксированного размера. Каждый раз, когда сервер принимает запрос клиента на соединение, создается серверный рабочий поток, и когда клиент прекращает работу, выполнение потока завершается.

- Сервер создает отдельный поток приема (accept thread), что позволяет основному потоку опрашивать глобальный флаг завершения работы, пока вызов accept остается заблокированным. Хотя сокеты и могут определяться как неблокирующиеся, потоки обеспечивают удобное универсальное решение. Следует отметить, что значительная часть расширенных функциональных возможностей Winsock призвана поддерживать асинхронные операции, тогда как потоки Windows дают возможность воспользоваться более простой и близкой к стандартам функциональностью синхронного режима работы сокетов.

- За счет некоторого усложнения программы усовершенствовано управление потоками, что позволило обеспечить поддержку состояний каждого потока.

- Данный сервер поддерживает также внутрипроцессные серверы (in-process servers), что достигается путем загрузки библиотеки DLL во время инициализации. Имя библиотеки DLL задается в командной строке, и серверный поток сначала пытается определить точку входа этой DLL. В случае успеха серверный поток вызывает точку входа DLL; в противном случае сервер создает процесс аналогично тому, как это делалось в программе serverNP. Пример DLL приведен в программе 12.3. Поскольку генерация исключений библиотекой DLL будет приводить к уничтожению всего серверного процесса, вызов функции DLL защищен простым обработчиком исключений.

При желании можно включить внутрипроцессные серверы и в программу serverNP. Самым большим преимуществом внутрипроцессных серверов является то, что они не требуют никакого контекстного переключения на другие процессы, в результате чего производительность может заметно улучшиться.

Поскольку в коде сервера использованы специфические для Windows возможности, в частности, возможности управления потоками и некоторые другие, он, в отличие от кода клиента, оказывается привязанным к Windows.

Программа 12.2. serverSK: сервер на основе сокета с внутрипроцессными серверами

```
/* Глава 12. Клиент-серверная система. ПРОГРАММА СЕРВЕРА.  
VERСИЯ НА ОСНОВЕ СОКЕТА. */
```

```
/* Выполняет указанную в запросе команду и возвращает ответ. */
```

```
/* Если удастся обнаружить точку входа разделяемой библиотеки,  
команды */
```

```
/* выполняются внутри процесса, в противном случае – вне процесса.
```

```
*/
```

```

/* ДОПОЛНИТЕЛЬНАЯ ВОЗМОЖНОСТЬ: argv [1] может содержать
имя библиотеки */
/* DLL, поддерживающей внутрипроцессные серверы. */

#define _NOEXCLUSIONS
#include "EvryThng.h"
#include "ClntSrvr.h" /* Определяет структуру записей запроса и ответа.
*/

struct sockaddr_in SrvSAddr;
/* Адресная структура сокета сервера. */
struct sockaddr_in ConnectSAddr; /* Подключенный сокет. */
WSADATA WStartData; /* Структура данных библиотеки сокета. */

typedef struct SERVER_ARG_TAG { /* Аргументы серверного потока.
*/
    volatile DWORD number;
    volatile SOCKET sock;
    volatile DWORD status;
    /* Пояснения содержатся в комментариях к основному потоку. */
    volatile HANDLE srv_thd;
    HINSTANCE dlhandle; /* Дескриптор разделяемой библиотеки. */
} SERVER_ARG;

volatile static ShutFlag = FALSE;
static SOCKET SrvSock, ConnectSock;
int _tmain(DWORD argc, LPCTSTR argv[]) {
    /* Прослушивающий и подключенный сокеты сервера. */
    BOOL Done = FALSE;
    DWORD ith, tstatus, ThId;
    SERVER_ARG srv_arg[MAX_CLIENTS];
    HANDLE hAcceptTh = NULL;
    HINSTANCE hDll = NULL;
    /* Инициализировать библиотеку WSA; задана версия 2.0, но будет
работать и версия 1.1. */
    WSASStartup(MAKEWORD(2, 0), &WStartData);
    /* Открыть динамическую библиотеку команд, если ее имя указано в
командной строке. */
    if (argc > 1) hDll = LoadLibrary(argv[1]);
    /* Инициализировать массив arg потока. */
    for (ith = 0; ith < MAXCLIENTS; ith++) {
        srv_arg[ith].number = ith;
        srv_arg[ith].status = 0;
        srv_arg[ith].sock = 0;
        srv_arg[ith].dlhandle = hDll;
    }
}

```

```

    srv_arg[ith].srv_thd = NULL;
}
/* Следовать стандартной процедуре вызова последовательности
функций socket/bind/listen/accept клиентом. */
SrvSock = socket(AF_INET, SOCK_STREAM, 0);
SrvSAddr.sin_family = AF_INET;
SrvSAddr.sin_addr.s_addr = htonl(INADDR_ANY);
SrvSAddr.sin_port = htons(SERVER_PORT);
bind(SrvSock, (struct sockaddr *)&SrvSAddr, sizeof SrvSAddr);
listen(SrvSock, MAX_CLIENTS);

/* Основной поток становится потоком
прослушивания/соединения/контроля. */
/* Найти пустую ячейку в массиве arg потока сервера. */
/* параметр состояния: 0 – ячейка свободна; 1 – поток остановлен; 2 —
поток выполняется; 3 – остановлена вся система. */
while (!ShutFlag) {
    for (ith = 0; ith < MAX_CLIENTS && !ShutFlag; ) {
        if (srv_arg[ith].status==1 || srv_arg[ith].status==3) { /* Выполнение
потока завершено либо обычным способом, либо по запросу останова. */
            WaitForSingleObject(srv_arg[ith].srv_thd INFINITE);
            CloseHandle(srv_arg[ith].srv_tnd);
            if (srv_arg[ith].status == 3) ShutFlag = TRUE;
            else srv_arg[ith].status = 0;
            /* Освободить ячейку данного потока. */
        }
        if (srv_arg[ith].status == 0 || ShutFlag) break;
        ith = (ith + 1) % MAXCLIENTS;
        if (ith == 0) Sleep(1000);
        /* Прервать цикл опроса. */
        /* Альтернативный вариант: использовать событие для генерации
сигнала, указывающего на освобождение ячейки. */
    }
    /* Ожидать попытки соединения через данный сокет. */
    /* Отдельный поток для опроса флага завершения ShutFlag. */
    hAcceptTh = (HANDLE)_beginthreadex(NULL, 0, AcceptTh,
&srv_arg[ith], 0, &ThId);
    while (!ShutFlag) {
        tstatus = WaitForSingleObject(hAcceptTh, CS_TIMEOUT);
        if (tstatus == WAIT_OBJECT_0) break; /* Соединение установлено. */
    }
    CloseHandle(hAcceptTh);
    hAcceptTh = NULL; /* Подготовиться к следующему соединению. */
}

```

```

    _tprintf(_T("Остановка сервера. Ожидание завершения всех потоков
сервера\n"));
    /* Завершить принимающий поток, если он все еще выполняется. */
    /* Более подробная информация об используемой логике завершения
*/
    /* работы приведена на Web-сайте книги. */
    if (hDll != NULL) FreeLibrary(hDll);
    if (hAcceptTh != NULL) TerminateThread(hAcceptTh, 0);
    /* Ожидать завершения всех активных потоков сервера. */
    for (ith = 0; ith < MAXCLIENTS; ith++) if (srv_arg [ith].status != 0) {
        WaitForSingleObject(srv_arg[ith].srv_thd, INFINITE);
        CloseHandle(srv_arg[ith].srv_thd);
    }
    shutdown(SrvSock, 2);
    closesocket(SrvSock);
    WSACleanup();
    return 0;
}

static DWORD WINAPI AcceptTh(SERVER_ARG * pThArg) {
    /* Принимающий поток, который предоставляет основному потоку
возможность опроса флага завершения. Кроме того, этот поток создает
серверный поток. */
    LONG AddrLen, ThId;
    AddrLen = sizeof(ConnectSAddr);
    pThArg->sock = accept(SrvSock, /* Это блокирующий вызов. */
        (struct sockaddr *)&ConnectSAddr, &AddrLen);
    /* Новое соединение. Создать серверный поток. */
    pThArg->status = 2;
    pThArg->srv_thd = (HANDLE)_beginthreadex (NULL, 0, Server, pThArg,
0, &ThId);
    return 0; /* Серверный поток продолжает выполняться. */
}

static DWORD WINAPI Server(SERVER_ARG * pThArg)
/* Функция серверного потока. Поток создается по требованию. */
{
    /* Каждый поток поддерживает в стеке собственные структуры данных
запроса, ответа и регистрационных записей. */
    /* ... Стандартные объявления из serverNP опущены ... */
    SOCKET ConnectSock;
    int Disconnect = 0, i;
    int (*dl_addr)(char *, char *);
    char *ws = " \0\t\n"; /* Пробелы. */
    GetStartupInfo(&StartInfoCh);

```

```

ConnectSock = pThArg->sock;
/* Создать имя временного файла. */
sprintf(TempFile, "%s%d%s", "ServerTemp", pThArg->number, ".tmp");
while (!Done && !ShutFlag) { /* Основной командный цикл. */
    Disconnect = ReceiveRequestMessage(&Request, ConnectSock);
    Done = Disconnect || (strcmp(Request.Record, "$Quit") == 0) ||
    (strcmp(Request.Record, "$ShutDownServer") == 0);
    if (Done) continue;
    /* Остановить этот поток по получении команды "$Quit" или
"$ShutDownServer". */
    hTrapFile = CreateFile(TempFile, GENERIC_READ |
GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, &TempSA,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    /* Проверка наличия этой команды в DLL. Для упрощения команды */
    /* разделяемой библиотеки имеют более высокий приоритет по
сравнению */
    /* с командами процесса. Прежде всего, необходимо извлечь имя
команды.*/
    i = strcspn(Request.Record, ws); /* Размер лексемы. */
    memcpy(sys_command, Request.Record, i);
    sys_command[i] = '\0';
    dl_addr = NULL; /* Будет установлен в случае успешного выполнения
функции GetProcAddress. */
    if (pThArg->dlhandle != NULL) { /* Проверка поддержки
"внутрипроцессного" сервера. */
        dl_addr = (int (*)(char *, char *))GetProcAddress(pThArg->dlhandle,
sys_command);
        if (dl_addr != NULL) __try {
            /* Защитить серверный процесс от исключений, возникающих в
DLL*/
            (*dl_addr)(Request.Record, TempFile);
        } __except (EXCEPTION_EXECUTE_HANDLER) {
            ReportError(_T("Исключение в DLL"), 0, FALSE);
        }
    }
    if (dl_addr == NULL) { /* Поддержка внутрипроцессного сервера
отсутствует. */
        /* Создать процесс для выполнения команды. */
        /* ... То же, что в serverNP ... */
    }
    /* ... То же, что в serverNP ... */
} /* Конец основного командного цикла. Получить следующую
команду. */
/* Конец командного цикла. Освободить ресурсы; выйти из потока. */
_tprintf(_T("Завершение работы сервера# %d\n"), pThArg->number);

```

```

shutdown(ConnectSock, 2);
closesocket(ConnectSock);
pThArg->status = 1;
if (strcmp(Request.Record, "$ShutDownServer") == 0) {
    pThArg->status = 3;
    ShutFlag = TRUE;
}
return pThArg->status;
}

```

### **Замечания по поводу безопасности**

В том виде, как она здесь представлена, данная клиент-серверная система не является безопасной. Если на вашей системе выполняется сервер и кому-то известен номер порта, через который вы работаете, и имя компьютера, то он может атаковать вашу систему. Другой пользователь, запустив клиентскую программу на своем компьютере, сможет выполнить на вашей системе команды, позволяющие, например, удалить или изменить файлы.

### **Пример: безопасная многопоточная DLL для обмена сообщениями через сокет**

Программа 12.4 представляет собой DLL, содержащую две функции для обработки символьных строк (в именах которых в данном случае присутствует "CS", от character string — строка символов), или потоковые функции сокета (socket streaming functions): SendCSMessage и ReceiveCSMessage, а также точку входа DllMain (см. главу 5). Указанные две функции играют ту же роль, что и функция ReceiveMessage, а также функции, использованные в программах 12.1 и 12.2, и фактически заменяют их.

Функция DllMain служит характерным примером решения проблемы долговременных состояний в многопоточной среде и объединяет TLS и библиотеки DLL.

Освобождать ресурсы при отсоединении потоков (случай DLL\_THREAD\_DETACH) особенно важно в случае серверной среды; если этого не делать, то ресурсы сервера, в конечном счете, исчерпаются, что может привести к сбоям в его работе или снижению производительности или к тому и другому одновременно.

### **Примечание**

*Некоторые из иллюстрируемых ниже концепций прямого отношения к сокетам не имеют, но, тем не менее, рассматриваются именно здесь, а не в предыдущих главах, поскольку данный пример предоставляет удобную возможность для иллюстрации методов создания безопасных многопоточных DLL в реалистических условиях.*

*Использующие эту DLL коды клиента и сервера, незначительно измененные по сравнению с программами 12.1 и 12.2, доступны на Web-сайте книги.*

Программа 12.4. SendReceiveSKST: безопасная многопоточная DLL

```

/* SendReceiveSKST.c — DLL многопоточного потокового сокета. */
/* В качестве разделителей сообщений используются символы конца */
/* строки ('\0'), так что размер сообщения заранее не известен. */
/* Поступающие данные буферизуются и сохраняются в промежутках
между */
/* вызовами функций. */
/* Для этой цели используются локальные области хранения потоков */
/* (Thread Local Storage, TLS), обеспечивающие каждый из потоков */
/* собственным закрытым "статическим хранилищем". */

#define _NOEXCLUSIONS
#include "EvryThng.h"
#include "ClntSrvr.h" /* Определяет записи запроса и ответа. */

typedef struct STATIC_BUF_T {
/* "static_buf" содержит "static_buf_len" байтов остаточных данных. */
/* Символы конца строки (нулевые символы) могут присутствовать, а
могут */
/* и не присутствовать. */
char static_buf[MAX_RQRS_LEN];
LONG32 static_buf_len;
} STATIC_BUF;

static DWORD TlsIx = 0; /* Индекс TLS – ДЛЯ КАЖДОГО ПРОЦЕССА
СВОЙ ИНДЕКС.*/
/* Для однопоточной библиотеки использовались бы следующие
определения:
static char static_buf [MAX_RQRS_LEN];
static LONG32 static_buf_len; */
/* Основная функция DLL. */

BOOL WINAPI DllMain(HINSTANCE hinstDLL, DWORD fdwReason,
LPVOID lpvReserved) {
    STATIC_BUF * pBuf;
    switch (fdwReason) {
    case DLL_PROCESS_ATTACH:
        TlsIx = TlsAlloc();
        /* Для основного потока подключение отсутствует, поэтому во время
        подключения процесса необходимо выполнить также операции по
        подключению потока. */
    case DLL_THREAD_ATTACH:
        /* Указать, что память не была распределена. */
        TlsSetValue(TlsIx, NULL);
        return TRUE; /* В действительности это значение игнорируется. */
    case DLL_PROCESS_DETACH:

```

```

/* Отсоединить также основной поток. */
pBuf = TlsGetValue(TlsIx);
if (pBuf != NULL) {
    free(pBuf);
    pBuf = NULL;
}
return TRUE;
case DLL_THREAD_DETACH:
    pBuf = TlsGetValue(TlsIx);
    if (pBuf != NULL) {
        free(pBuf);
        pBuf = NULL;
    }
    return TRUE;
}
}

_declspec(dllexport)
BOOL ReceiveCSMessage(REQUEST *pRequest, SOCKET sd) {
    /* Возвращаемое значение TRUE указывает на ошибку или
отсоединение. */
    BOOL Disconnect = FALSE;
    LONG32 nRemainRecv = 0, nXfer, k; /* Должны быть целыми со
знаком. */
    LPSTR pBuffer, message;
    CHAR TempBuf[MAX_RQRS_LEN + 1];
    STATIC_BUF *p;
    p = (STATIC_BUF *)TlsGetValue(TlsIx);
    if (p == NULL) { /* Инициализация при первом вызове. */
        /* Распределять это хранилище будут только те потоки, которым оно
*/
        /* необходимо. Другие типы потоков могут использовать TLS для
иных целей. */
        p = malloc(sizeof(STATIC_BUF));
        TlsSetValue(TlsIx, p);
        if (p == NULL) return TRUE; /* Ошибка. */
        p->static_buf_len = 0; /* Инициализировать состояние. */
    }
    message = pRequest->Record;
    /* Считать до символа новой строки, оставляя остаточные данные в
статическом буфере. */
    for (k = 0; k < p->static_buf_len && p->static_buf[k] != '\0'; k++) {
        message[k] = p->static_buf[k];
    } /* k – количество переданных символов. */

```



```

        if (k < p->static_buf_len) { /* В статическом буфере обнаружен нулевой
СИМВОЛ. */
            message[k] = '\0';
            p->static_buf_len -= (k + 1); /* Скорректировать состояние
статического буфера. */
            memcpy(p->static_buf, &(p->static_buf[k + 1]), p->static_buf_len);
            return FALSE; /* Входные данные сокета не требуются. */
        }

        /* Передан весь статический буфер. Признак конца строки не
обнаружен.*/
        nRemainRecv = sizeof(TempBuf) - 1 - p->static_buf_len;
        pBuffer = message + p->static_buf_len;
        p->static_buf_len = 0;
        while (nRemainRecv > 0 && !Disconnect) {
            nXfer = recv(sd, TempBuf, nRemainRecv, 0);
            if (nXfer <= 0) {
                Disconnect = TRUE;
                continue;
            }
            nRemainRecv -= nXfer;
            /* Передать в целевое сообщение все символы вплоть до нулевого,
если таковой имеется. */
            for (k = 0; k < nXfer && TempBuf[k] != '\0'; k++) {
                *pBuffer = TempBuf[k];
                pBuffer++;
            }
            if (k >= nXfer) { /*Признак конца строки не обнаружен, читать
дальше*/
                nRemainRecv -= nXfer;
            } else { /* Обнаружен признак конца строки. */
                *pBuffer = '\0';
                nRemainRecv = 0;
                memcpy(p->static_buf, &TempBuf[k + 1], nXfer - k - 1);
                p->static_buf_len = nXfer - k - 1;
            }
        }
        return Disconnect;
    }

_declspec(dllexport)
BOOL SendCSMessage(RESPONSE *pResponse, SOCKET sd) {
    /* Послать запрос серверу в сокет sd. */
    BOOL Disconnect = FALSE;
    LONG32 nRemainSend, nXfer;

```

```

LPSTR pBuffer;
pBuffer = pResponse->Record;
nRemainSend = strlen(pBuffer) + 1;
while (nRemainSend > 0 && !Disconnect) {
    /* Отправка еще не гарантирует, что будет отослано все сообщение. */
    nXfer = send(sd, pBuffer, nRemainSend, 0);
    if (nXfer <= 0) {
        fprintf(stderr, "\nОтключение сервера до посылки запроса
завершения");
        Disconnect = TRUE;
    }
    nRemainSend -= nXfer;
    pBuffer += nXfer;
}
return Disconnect;
}

```

### **Комментарии по поводу DLL и безопасной многопоточной среды**

- Всякий раз, когда создается новый поток, вызывается функция DllMain с опцией DLL\_THREAD\_ATTACH, но для основного потока отдельного вызова с опцией DLL\_THREAD\_ATTACH не существует. В случае основного потока должна использоваться опция DLL\_PROCESS\_ATTACH.

- Вообще говоря, в том числе и в данном случае (возьмите, например, поток, принимающий сообщения (accept thread)), некоторым потокам распределение памяти может и не требоваться, но DllMain не в состоянии различать отдельные типы потоков. Поэтому на участке кода, соответствующем варианту выбора DLL\_THREAD\_ATTACH, фактического распределения памяти не происходит; здесь только инициализируется параметр TLS. Распределение памяти осуществляется точкой входа ReceiveCSMessage при первом ее вызове. Благодаря этому собственная память выделяется только тем потокам, которые в этом действительно нуждаются, и различные типы потоков получают ровно столько ресурсов, сколько им требуется.

- Хотя рассматриваемая библиотека DLL и обеспечивает безопасную многопоточную поддержку, любой поток в каждый момент времени может работать только с одним сокетом, поскольку долговременные состояния ассоциируются не с сокетами, а с потоками. Этот момент учитывается в следующем примере.

- Исходным кодом DLL, размещенным на Web-сайте, предусмотрен вывод общего количества вызовов DllMain в соответствии с их типами.

- Даже при таком решении существует риск утечки ресурсов. Некоторые потоки, например поток приема сообщений, могут вообще не завершаться, и поэтому не будут отсоединены от библиотеки DLL. Для остающихся активных потоков функция ExitProcess вызовет DllMain с опцией DLL\_PROCESS\_DETACH, а не DLL\_THREAD\_DETACH. В

данном случае никаких проблем не возникает, поскольку поток приема сообщений никаких ресурсов не распределяет, а освобождение памяти происходит по завершении процесса. Однако, проблемы возможны в тех случаях, когда потоки распределяют такие ресурсы, как временные файлы. Поэтому окончательное решение должно предусматривать создание глобально доступного списка ресурсов. Тогда участок кода, соответствующий опции DLL\_PROCESS\_DETACH, мог бы взять на себя просмотр этого списка и освобождение ненужных ресурсов.

**Задание 2.** Воспроизведите текст программы. Выполните его.

**Задание 3.** Подготовить отчет о выполненной работе

**Критерии оценивания:**

Отметка	Объем выполнения работы в %
«5» (отлично)	90 – 100
«4» (хорошо)	70 – 89
«3» (удовлетворительно)	50 – 69
«2» (неудовлетворительно)	менее 50

### 3. КРИТЕРИИ ОЦЕНИВАНИЯ ВЫПОЛНЕННЫХ РАБОТ

Оценка за практическую работу складывается из оценки за выполнение работы и оценки за защиту.

**Оценка «отлично»** ставится, если студент выполнил практическую работу в полном объеме с соблюдением необходимой последовательности действий; в отчете правильно и аккуратно выполняет все записи, таблицы, рисунки, чертежи, графики, вычисления; правильно выполняет анализ ошибок.

**На защите студент при ответе на вопросы** правильно понимает сущность вопроса, дает точное определение и истолкование основных понятий; сопровождает ответ новыми примерами, умеет применить знания в новой ситуации; может установить связь между изучаемым и ранее изученным материалом, а также с материалом, усвоенным при изучении других дисциплин.

**Оценка «хорошо»** ставится, если студент выполнил требования к оценке "5", но допущены 2-3 недочета.

**На защите студент при ответе на вопросы** ответ студента удовлетворяет основным требованиям к ответу на оценку 5, но дан без применения знаний в новой ситуации, без использования связей с ранее изученным материалом и материалом, усвоенным при изучении других дисциплин; студент допустил одну ошибку или не более двух недочетов и может их исправить самостоятельно или с небольшой помощью преподавателя.

**Оценка «удовлетворительно»** ставится, если студент выполнил работу не полностью, но не менее 50% объема практической работы, что позволяет получить правильные результаты и выводы; в ходе проведения работы были допущены ошибки.

**На защите студент при ответе на вопросы** правильно понимает сущность вопроса, но в ответе имеются отдельные пробелы в усвоении вопросов курса, не препятствующие дальнейшему усвоению программного материала; допустил не более одной грубой ошибки и двух недочетов.

**Оценка «неудовлетворительно»** ставится, если студент выполнил работу не полностью или объем выполненной части работы не позволяет сделать правильных выводов;

**На защите студент при ответе на вопросы** не овладел основными знаниями и умениями в соответствии с требованиями программы и допустил

больше ошибок и недочетов, чем необходимо для оценки 3 или не может ответить ни на один из поставленных вопросов.

## **4. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКИХ РАБОТ**

**Информационное обеспечение реализации программы**  
**Перечень используемых учебных изданий, Интернет-ресурсов, дополнительной литературы**

### **4.1 Печатные издания**

### **4.2 Электронные издания (электронные ресурсы)**

1 Гниденко, И. Г. Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2021. — 235 с. — (Профессиональное образование). — ISBN 978-5-534-05047-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/472502>

2. Подбельский, В. В. Программирование. Базовый курс C# : учебник для среднего профессионального образования / В. В. Подбельский. — Москва : Издательство Юрайт, 2020. — 369 с. — (Профессиональное образование). — ISBN 978-5-534-11467-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/456697>

### **Интернет-ресурсы**

1. Образовательный портал INTUIT.RU <http://www.intuit.ru>
2. METANIT.COM. Сайт о программировании <https://metanit.com>
3. Журнал «Моя профессиональная карьера» - Режим доступа: <https://www.elibrary.ru/item.asp?id=45669781>
4. Журнал «Апробация» - Режим доступа <https://www.elibrary.ru/item.asp?id=23216950>
5. Журнал «Новая наука и стратегия развития» - Режим доступа: <https://www.elibrary.ru/item.asp?id=27424248>
6. Журнал «Вестник сыктывкарского университета. Серия 1: Математика. Механика. Информатика» <https://www.elibrary.ru/item.asp?id=32546230>

### **Электронно-библиотечные системы:**

Доступ авторизованных пользователей через Интернет

ЭБС «IPRbooks», ООО «Ай Пи Эр Медиа»

ЭБС «Электронная библиотека технического вуза», ООО «Политехресурс»

ЭБС «Лань», ООО «Издательство Лань»  
ЭБС «elibrary», ООО «РУНЭБ»  
ЭБС «ЮРАЙТ»  
ЭБС «Book.ru»