Федеральное агентство по образованию РФ
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО «АмГУ»)

УТВЕРЖДАЮ
Зав. кафедрой ИУС
_____А.В.Бушманов
«_____»_____

ИНФОРМАТИКА

Учебно-методический комплекс

для студентов специальности 220302 – Автоматизация технологических процессов и производств

Составители доцент кафедры ИУС Галаган Т.А.,
доцент кафедры ОМиИ Макарчук Т.А.

Факультет математики и информатики

Кафедра информационных и управляющих систем

2007

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

Т.А. Галаган, Т.А. Макарчук

Учебно-методический комплекс по дисциплине «Информатика». Для студентов специальности 220302 – «Автоматизация технологических процессов и производств» очной формы обучения.- Благовещенск: Амурский гос. ун-т, 2007.

Пособие содержит рабочую программу, курс лекций, методические рекомендации по проведению и выполнению лабораторных работ. Составлено в соответствии с требования государственного образовательного стандарта.

I. ПРИМЕРНАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ, УТВЕРЖДЕННАЯ МИНИСТЕРСТВОМ ОБРАЗОВАНИЯ РФ

Государственный образовательный стандарт высшего профессионального образования

Направление подготовки дипломированного специалиста 654600 - Информатика и вычислительная техника

Образовательная программа – 22030 – Автоматизация технологических процессов и производств

Наименование дисциплины – Информатика

Блок естественно-научных дисциплин, индекс ЕН.Ф.02.

Всего часов - 200

Содержание разделов:

Основные понятия информатики: информация, информационная технология, участники процесса обработки информации; компьютер как техническое средство реализации технологий, структура компьютера и программного обеспечения с точки зрения конечного пользователя, средства и алгоритмы представления, хранения и обработки текстовой и числовой информации; среды конечного пользователя; организация и средства человеко-машинного интерфейса, мультисреды и гиперсреды; назначение и основы использования систем искусственного интеллекта; понятие о сетях ЭВМ, информационных технологиях на сетях; основы телекоммуникаций и распределенной обработки данных; понятие об экономических и правовых аспектах информационных технологий; основы защиты информации и сведений, составляющих государственную тайну; методы защиты информации; компьютерный практикум.

II. РАБОЧАЯ ПРОГРАММА

Курс 1 семестр 1, 2

Лекции 72 (час.) Экзамен 2 семестр

Практические (семинарские) занятия - (час.) Зачет 1 семестр

Лабораторные занятия 72 (час.)

Самостоятельная работа 66 (час.)

Всего часов 200 час.

1. Цели и задачи учебной дисциплины «Информатика», ее место в учебном процессе.

1.1. Место дисциплины в учебном процессе

Предлагаемый курс обеспечивает базовую подготовку студентов в области использования средств вычислительной техники: для всех курсов, использующих автоматизированные методы анализа и расчетов, курсового и дипломного проектирование.

1.2. Цели преподавания учебной дисциплины «Информатика»:

- воспитание у студентов информационной культуры;
- обучение студентов теоретическим основам и практическим навыкам работы с аппаратным и программным обеспечением компьютера

1.3. Задачи изучения дисциплины:

- углубить знания студентов по основному аппаратному обеспечению и периферийным устройствам компьютера;
- научить студентов решать задачи, возникающие в процессе сопровождения и эксплуатации программных средств;
- освоить современные методы и средства программирования, этапы разработки программного обеспечения;
- ознакомить студентов с принципами представления данных и функционирования информационных систем.

1.4. Перечень учебных курсов, освоение которых необходимо для изучения дисциплины «Информатика»

Для успешного освоения курса необходимы знания курса "Информатика" в объеме средней общеобразовательной школы.

1.5. После изучения дисциплины студенты должны знать и уметь использовать:

- операционной системой Windows 2000/XP, Windows NT;
- операционными оболочками Total Commander, Far Manager;
- служебными программами Windows;

- архиваторами WInRar, WinZip;
- прикладные программы Microsoft Office: Word, Excel, Access, Power Point;
- программой-обозревателем Internet Explorer;
- системой программирования C++.

2. Темы дисциплины и их содержание.

1. Общие характеристики процессов сбора, передачи, обработки и накопления информации.

Понятие информации. Знания и данные. Форма представления информации. Виды информации. Свойства информации. Позиционные системы счисления информации. Общая структурная схема информационного процесса. Информационные системы и технологии. Информационное общество. Понятие системы счисления. Классификация систем счисления. Двоичная система счисления. Правила перевода из десятичной в двоичную систему. Правила перевода из двоичной в десятичную систему счисления. Системы счисления, родственные двоичной (восьмеричная, шестнадцатеричная).

2. Технические средства реализации информационного процесса.

История развития вычислительной техники. Классы ЭВМ и их основные характеристики. Основные блоки ПК и их назначение. Процессор и его характеристики: разрядность, тактовая частота, быстродействие. Запоминающие устройства. Периферийные устройства: монитор, клавиатура, принтер, сканер, модем, графопостроитель, дигитайзер, манипуляторы, средства мультимедиа.

3. Программные средства реализации информационных процессов.

Классификация программных средств. Операционная система: понятие, составные части, классификация. Физическая организация данных на носителях, файловые системы: FAT, NTFS, WinFS. Операционные оболочки. Сервисные программные средства: форматирование, дефрагментация, проверка диска, очистка диска, сведения о системе. Архивация данных. Краткий обзор современных программных средств.

4. Локальные сети ЭВМ.

Понятие компьютерной сети. Устройства сети: сервер, рабочая станция, коммуникационные узлы. Характеристики сети. Классификация сетей по территориальному признаку: LAN, MAN, WAN сети. Эталонная модель OSI. Топология компьютерных сетей. Линии связи. Стек протоколов TCP/IP. Программное обеспечение компьютерных сетей: одноранговые и с выделенным сервером. Сетевые ОС, Windows NT, Novell NetMare.

5. Глобальные компьютерные сети.

Понятие Интернет. История создания сети Интернет. Современная структура сети Интернет. Адресация в сети Интернет: IP, URL. Основные протоколы сети Интернет: http, ftp, telnet, SMTP, POP, IMAP, MIME. Электронная почта. Word Wide Web: URL-адрес, гипертекст, программы-браузеры, языки разметки гипертекста HTML и XML. Поиск информации в Интернет. UseNet, Chat, IP-телефония.

6. Защита информации в сети.

Основы защиты информации и сведений, составляющих государственную тайну: угрозы информации в сети, основные аспекты безопасности. Методы защиты информации: криптография, электронная подпись, аутентификация, сертификация Web-узлов. Компьютерные вирусы и антивирусные программные средства.

7. Базы данных.

Понятие базы данных. Модели организации данных. Язык SQL. Системы управления базами данных. Основные понятия СУБД Access: поле данных, ключ поля данных, схема данных, таблицы, формы, запросы, отчеты.

8. Алгоритмизация и программирование.

Понятие алгоритма. Свойства алгоритма. Блок-схема. Основные конструкции алгоритмов. Понятие программы. Этапы разработки программ: определение исходных данных, выбор метода решения, алгоритмизация, программирование, отладка и тестирование.

9. Программное обеспечение программирования.

Понятие языка программирования. Языки программирования низкого и высокого уровней. Обзор языков программирования. Программы-трансляторы. Системы программирования.

10. Технология программирования на языке высокого уровня.

Структурное, модульное, объектно-ориентированное программирование. Основные понятия языка: идентификатор, оператор, ключевое слово. Структура программных объектов (подпрограмм, модуля, программы). Стандартные типы данных языка. Арифметические операции, выражения и функции. Операторы: присваивание, полное и неполное ветвление, выбор, цикл с параметром, с предусловием, с постусловием. Вектора и матрицы.

3. Тематическое планирование лабораторных занятий по курсу "Информатика". Компьютерный практикум.

1. Операционная система Windows.

Папка "Мой компьютер". Работа с файлами, папками, ярлыками: создание, открытие, переименование, копирование, перемещение, удаление. Работа с деревом каталогов (файловой структурой) в программе "Проводник". Восстановление файлов с помощью программы "Корзина". Виды меню: системное, строка меню, контекстное, главное, панель инструментов. Элементы интерфейса: радиокнопки, флажки, командные кнопки. Основы работы со стандартными программами Windows: Блокнот, калькулятор, графический редактор Paint. Панель управления. Поиск файлов и папок. Настройка интерфейса Windows, в том числе настройка главного меню.

2. Операционные оболочки Far Manager, Total Commander.

Работа с панелями информационного окна, управление пакетом с помощью функциональных клавиш и ниспадающего меню. Создание, открытие, просмотр, копирование, перемещение, переименование, удаление файлов и каталогов. Работа с архивными файлами.

3. Сервисное программное обеспечение Windows.

Программы обслуживания дисков, сведения о системе, программы-архиваторы (WinZip, WinRar), антивирусные программы, программы обслуживания сети. Запись и чтение информации в локальной сети.

4. Текстовый процессор Word.

Классификация текстовых редакторов: простейшие текстовые редакторы, текстовые процессоры, издательские системы. Параметры страницы. Форматирование абзаца, опции шрифта, создание списка. Колонтитулы. Вставка объектов (Eqution, WordArt, ClipArt и др.). Создание ссылок (концевые сноски и оглавление). Оформление таблиц и вычисления в них. Проверка орфографии.

5. Электронная таблица Excel.

Назначение Excel. Адресация ячеек. Форматирование ячеек. Создание и копирование формул. Математические, статистические и логические функции. Сортировка и фильтрация данных. Работа с базами данных в Excel. Графический анализ данных: гистограмма, линейный график, круговая диаграмма.

6. Система управления базами данных ACCESS.

Построение структура реляционной базы данных: определение полей, ключевого поля, тип и формат полей. Режим конструктора. Объекты БД: таблица, форма, запрос. Сортировка и фильтрация данных. Схема данных: связи между таблицами $1\leftrightarrow1$, $1\leftrightarrow\infty$, $\infty\leftrightarrow\infty$. Построение простых запросов.

7. Компьютерная графика CorelDRAW, Adobe PhotoShop.

Создание графических изображений, системы цветов и форматов графических файлов. Создания рекламы, коллажей и многого другого. Работа со слоями и цветокоррекция изображений, работа с каналами и масками.

8. Презентации PowerPoint

Рисунки и графические примитивы на слайдах. Выбор дизайна презентации. Редактирование и сортировка слайдов. Использование анимации в презентации. Интерактивная презентация – переходы между слайдами. Демонстрация презентации.

9. Знакомство с C++. Технология программирования задач линейной структуры

Составление программы вычисления значения выражения, типа:
$\sqrt{x-1} + 1/(x-3)$.

10. Технология программирования задач разветвляющейся структуры

Написать программу, вычисляющую f(x) на промежутке.

11. Технология программирования задач с множественным выбором.

Программы: по выбранному элементу списка выдается соответствующая информация; по введенному значению выделяется соответствующая строка списка.

12. Технология программирования задач циклической структуры

Циклы с параметром, с предусловием, с постусловием. Составление программ вычисления суммы, произведения, среднего арифметическое членов числового ряда. Программы вычисление с заданной точностью ε суммы ряда. Написание программы табулирования функции.

13.Технология программирования задач с данными типа одномерный массив (вектор)

Описание данных типа одномерный массив. Написание программ, в которых для одномерного массива осуществляется: задание элементов массива (псевдослучайно, по определенному правилу, пользователем); нахождение суммы (произведения) элементов массива; определение количества элементов, соответствующих заданному условию; нахождение максимального (минимального) элемента массива; сортировка элементов вектора.

14.Технология программирования задач с данными типа матрица

Описание данных типа двумерный массив. Написание программ, в которых для двумерного массива осуществляется: нахождение суммы (произведения) строк, столбцов матрицы; определение количества элементов n-строки, соответствующих заданному условию; нахождение максимального (минимального) элемента n-столбца матрицы; получение новой матрицы.

15. Функции

Выделение подпрограмм. Вызов функции в программе. Формальные, фактические параметры

16. Символьные строки

Описание данных символьный массив, инициализация, использование библиотеки string.h

17. Указатели

Объявление и инициализация указателей. Связь массивов и указателей. Передача массивов в качестве параметров в функции.

18. Ссылки

Передача значений из функций по ссылке.

4. Распределение времени по курсу "Информатика"

Распределение часов на лекционные занятия

Тема	**Кол-во часов**
1 семестр	
Общие характеристики процессов сбора, передачи, обработки и накопления информации	2
Технические средства реализации информационных процессов	2
Программные средства реализации информационных процессов	2
Локальные сети ЭВМ	2
Глобальные компьютерные сети	2
Защита информации в сети	2
Компьютерная графика	2
Базы данных	2
Моделирование решения функциональных и вычислительных задач	2
ВСЕГО	18
2 семестр	
Алгоритмизация и программирование	4
Программное обеспечение программирования	4
Технология программирования на языке высокого уровня	10
ВСЕГО	18

Распределение часов на лабораторные занятия

Тема	Кол-во часов
1 семестр	
Операционная система Windows	4
Операционные оболочки Far Manager, Total Commander	2
Сервисное программное обеспечение Windows	2
Текстовый процессор Word	4
Презентации PowerPoint	2
Компьютерная графика	4
ВСЕГО	18
2 семестр	
Электронная таблица Excel	6
Система управления базами данных ACCESS	6
Знакомство со средой программирования С++	2
Технология программирования задач линейной структуры	2
Программирование задач разветвляющейся структуры	4
Программирование задач с множественным выбором	2
Программирование задач циклической структуры	6
Программирование задач с данными типа вектор	4
Технология программирования задач с данными типа матрица	4
Создание программы с выделением пользовательских функций	4
Использование указателей	6
Передача значений из функций по ссылке	4
Системы счисления	2
ВСЕГО	36

Вопросы для самостоятельной работы

Тема	Кол-во часов, специальность
1 семестр	
Работа с файлами, папками, ярлыками в ОС Windows 95/98/2000.	4
Работа с файлами и каталогами в операционной системе Ms-Dos	4
Назначение функциональных клавиш в Norton-подобных оболочках	2
Электронная почта в Интернет. Поисковые системы	6
Создание Web-страницы. Гипертекстовые документы	8
Организация базы данных в Excel	8
2 семестр	
Моделирование решения вычислительных и функциональных задач с применением линейной структуры	6
Моделирование решения задач с применением раз-	

ветвляющей структуры	6
Моделирование решения задач с применением оператора выбора	4
Моделирование решения задач с применением циклической структуры	6
Моделирование решения задач со сложной структурой данных типа массив	6
Моделирование решения задач со сложной структурой	6
ВСЕГО	**66**

3. Перечень промежуточных форм контроля знаний студентов.

По данному курсу предполагается оценка знаний по каждой теме лабораторной работы с использованием тестов, контрольных заданий, контрольных вопросов. Для оценки знаний по темам самостоятельной работы выполняются индивидуальные домашние задания.

3.1. Вопросы к зачету (1 семестр)

1. Информатика как наука. Характеристика информационного общества.
2. История развития вычислительной техники.
3. Информация и данные. Свойства информации. Единицы информации. Кодирование, декодирование.
4. Основное аппаратное обеспечение ПК. Материнская плата. Процессор. Характеристики процессора: быстродействие (производительность), разрядность, тактовая частота.
5. Классификация запоминающих устройств: внутренние и внешние ЗУ. Емкость ЗУ.
6. Классификация программного обеспечения ПК.
7. Базовое программное обеспечение ПК. Понятия ОС- операционной системы, ОО – операционной оболочки. Базовые элементы ОС. Классификация ОС.
8. Сервисное программное обеспечение ПК (форматирование, дефрагментация, архивация, программы обслуживания диска).
9. Компьютерные вирусы, их классификация. Антивирусные программы, их функции (детектор, доктор, ревизор, сторож, вакцинация).
10. Файловая система диска: кластер, люфт, Fat 16, Fat 32, NTFS. Понятия файл, каталог, директория, ярлык, их атрибуты.
11. Операционные оболочки: примеры, путь к ОО. Назначение функциональных клавиш.
12. Операционная система Windows. Характеристики Windows (6-7 характеристик). Виды меню в Windows. Понятия: Мой компьютер, Корзина, Панель управления, папки Windows, Мои документы, Program Files.
13. Компьютерные сети: виды устройств (сервер, рабочая станция, повторитель, коммутатор, мост, шлюз), типы передающей среды, топология локальной сети. Классификация сетей по территориальному признаку (LAN, MAN, WAN), по распределению сетевой ОС (сети с выделенным сервером, одноранговые сети). Сетевые операционные системы (примеры).

14. Защита информации. Три аспекта безопасности. Методы защиты: аутентификация, электронная подпись, криптография, сертификация.
15. Интернет. Службы Интернет (WWW, электронная почта, FTP, chat, телеконференция). Гипертекстовые документы. Программы-браузеры. Протокол IP/TCP. IP-адрес компьютера. URL-адрес ресурса.
16. Классификация текстовых редакторов: простейшие текстовые редакторы, текстовые процессоры, издательские системы. Команды Word: Параметры страницы, Форматирование абзаца, шрифта, списка, колонтитулов. Объекты (wordArt, ClipArt, Equation). Таблица. Проверка правописания.
17. Электронная таблица Excel. Виды адресации ячеек (относительная, абсолютная, смешанная). Форматирование ячеек (типы данных ячеек, границы, шрифт, заливка). Виды диаграмм. Функции СУММ, СРЗНАЧ, МАХ, МИН, ЕСЛИ, И, ИЛИ, НЕ.
18. Компьютерная графика (растровая, векторная, фрактальная).

3.2. Вопросы к экзамену (2 семестр).

1. Студент допускается к экзамену при условии посещения всех лекционных и лабораторных занятий.

1. Алгоритм. Свойства алгоритма.
2. Этапы создания программы
3. Понятия язык и система программирования
4. Данные, типы данных языка С++
5. Арифметические операции и библиотечные функции
6. Операторы ввода-вывода
7. Структура программы языка С++
8. Условный оператор
9. Оператор выбора
10. Цикл: понятие, блок-схема цикла с пред- и постусловиями, оператор цикла с параметром.
11. одномерные массивы
12. Двумерные массивы. Обращение к элементам случайным образом, по строкам, по столбцам.
13. Указатели. Операции с указателями.
14. Связь массивов и указателей. Передача массива в функцию
15. Символьные массивы. Библиотечные функции работы со строками.
16. Создание пользовательский функций. Формальные и фактические параметры. Вызов функции
17. Ссылки. Передача значений из функций по ссылке.
18. База данных. Модели базы данных. Основные элементы и объекты базы данных. Типы связей

4. Учебно-методические материалы.

4.1. Основная литература.

1. **Информатика. Базовый курс** [Текст]: учеб.: рек. Мин. обр. РФ / Ред. С.В.

Симонович. - СПб.: Питер, 2000, 2004,2005,2006. - 638 с.

2. **Информатика** [Текст]: учеб.: рек. Мин. обр. РФ/ под ред. Н. В. Макаровой. - М.: Финансы и статистика, 2000, 2005, 2001. - 268 с.

3. **Информатика** [Текст]: практикум по технологии работы на компьютере: рек. Мин. обр. РФ / под ред. Н. В. Макаровой. - М.: Финансы и статистика, 2005. - 256 с.

4. **Могилев, А. В.** Информатика [Текст]: УЧЛ - Учебное пособие / А. В. Могилев, Н. И. Пак, Е. К. Хеннер / под ред. Е. К. Хеннер. - М.: Академия, 2004. - 842 с.

5. **Бобровский, С. И.** Delphi 7 [Текст]: Учебный курс / С. И. Бобровский, 2004, 2006. - 736 с.

4.2. Дополнительная литература

1. **Гиляревский, Р. С.** Основы информатики [Текст]: Курс лекций / Р. С. Гиляревский. - М.: Экзамен, 2003. - 320 с.

2. **Информатика для юристов** и экономистов [Текст]: Учебник / Ред. С.В. Симонович. - СПб. : Питер, 2001, 2003. - 688 с.

3. **Ляхович, В. Ф.** Основы информатики [Текст]: учеб. пособие: Рек. Мин. обр. РФ / В.Ф. Ляхович, С.О. Крамаров. - 3-е изд. - Ростов н/Д : Феникс, 2003. - 700 с.

4. **Могилев, А. В.** Практикум по информатике [Текст]: учеб. пособие / А.В. Могилев, Н.И. Пак, Е.К. Хеннер; Под ред. Е.К. Хеннера. - М. : Академия, 2002. - 608 с.

5. **Острейковский, В. А.** Информатика [Текст]: Учеб.: Рек. Мин. обр. РФ / В.А. Острейковский. - М. : Высш. шк., 1999. - 512 с.

6. **Галаган Т.А.** Алгоритмические языки и программирование. Язык С++. Курс лекций. Рек. ДВ РУМЦ. – Благовещенск: Изд-во АмГУ. 2007. 147 с.

7. **Галаган Т.А., Соловцова Л.А.** Язык программирование С++ в примерах и задачах. Практикум. Благовещенск: Изд-во АмГУ, 2005. 104 с.

8. **Павловская Т.А.** С/С++. Программирование на языке высокого уровня. СПб.: Питер, 2004. 461 с. (Допущено МО РФ)

9. **Павловская Т.А., Щупак Ю.А.** С/С++. Структурное программирование. Практикум. СПб.: Питер, 2004. 239 с.

III. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОМУ СОСТАВУ

1. Методические рекомендации по проведению лекционных занятий

Задача лекции состоит не столько в изложении системы теоретических знаний, сколько в общении с аудиторией, сообщении ей смысла и значения излагаемого материала, в дальнейшем развитии знания. Полученные в ходе лекции знания, часто носят характер поверхностного усвоения, при этом должны служить дальнейшим мотивом и основой для дальнейшей организации самостоятельной учебно-познавательской деятельности по приобретению новых знанию и умений, приводящей к глубокому освоению понятий, как отдельной темы, так и науки в целом.

На очном лекционном занятии в вводной части определяются минимальные

знаний, умения и навыки, подлежащие усвоению в ходе изучения темы курса. В основной части рассматривается довольно большой объем материала, в основном, обзорного характера. В заключительной части лекции излагается постановка типовых задач темы, решение которых подробно будет рассмотрено на лабораторных занятиях.

Изложение информационного материала лекции предполагает использование объяснительно-иллюстративного метода с применением фронтальной формы организации обучения.

2. Методические рекомендации по проведению лабораторных работ

На лабораторных занятиях по информатике формируется и совершенствуется практический уровень владения информационными процессами, основывающийся на применении теоретических знаний. Для проведения лабораторных занятий со студентами по дисциплине «Информатика» используются компьютерные классы. Занятия в компьютерном классе предполагают индивидуальную или парно-групповую формы организации обучения.

Этапы проведения лабораторной работы следующие:

- Контрольный опрос студентов для проверки готовности к выполнению лабораторной работы (до 10 мин).
- Выдача индивидуального задания и пояснения о порядке выполнения индивидуального задания (до 5 мин).
- Выполнение индивидуального задания (около 1 ч.)
- Оформление результатов работы. Сдача выполненной работы преподавателю (до 10 мин).
- Получение домашнего задания (1-2 мин.)
- Приведение в порядок рабочего места, в том числе закрытие всех рабочий окон и уничтожение созданных на винчестере индивидуальных файлов (3-4 мин).

Индивидуальные задания для лабораторных работ должны быть представлены конкретно-практическими и творческими задачами.

На первой ступени изучения темы выполняются конкретно-практические задачи, при решении которых формируется минимальный набор умений. Преподаватель опосредованно руководит самостоятельной познавательной деятельностью студентов, консультирует студентов при возникновении непосильных затруднений в ходе решения задачи, обращает внимание группы "опасные" места решения. Отработка минимального набора навыков завершается во внеаудиторное время при выполнение домашней работы. Принимая во внимание сложность доступа некоторыми студентами к компьютерной техники во внеаудиторное время, домашние задания по "Информатике" должны носит большей части моделирующий характер.

Вторая ступень изучения темы дифференцируется в зависимости от степени усвоения его обязательного уровня. Студенты, овладев основами теории и усвоив содержание типовых методов и приемов решения задач, приступают к решению творческих задач. Если уровень знаний и умений, демонстрируемых студентом при контрольном обследовании, не соответствует установленным требованиям,

студент вновь возвращается к стандартным упражнениям, но под более пристальным наблюдением преподавателя.

По завершению изучения отдельной темы курса по результатам выполнения лабораторных работ каждый студент получает оценку.

Студенты, пропустившие лабораторные занятия, должны их выполнить во внеаудиторное время и отчитаться до начала зачетно-экзаменационной сессии.

3. Методические рекомендации по организации контроля знаний студентов

В Университете качество освоения образовательных программ оценивается путем осуществления текущего контроля успеваемости, проведения промежуточных аттестаций и итогового контроля по окончании семестра.

На первом занятии до сведения студентов доводятся требования и критерии оценки знаний по дисциплине.

Целью текущего контроля успеваемости является оценка качества освоения студентами образовательных программ в течение всего периода обучения. К главной задаче текущего контроля относится повышение мотивации студентов к регулярной учебной работе, самостоятельной работе, углублению знаний, дифференциации итоговой оценки знаний.

Текущий контроль успеваемости осуществляется систематически и, как правило, преподавателем, ведущим лабораторные занятия. Формами текущего контроля являются письменные опросы, автоматизированное тестирование, аудиторные контрольные работы, отчеты по лабораторным работам, домашние задания. В течение семестра преподавателем должно быть проведено не менее 7-ми контрольных проверок знаний по каждому студенту из учебной группы.

Результаты текущего контроля служат основанием для прохождения студентом промежуточной аттестации.

Итоговый контроль (зачет или экзамен) по информатике преследуют цель оценить работу студентов за курс, полученные теоретические знания, их прочность, развитие творческого мышления, навыки самостоятельной работы, умение синтезировать полученные знания и применять их при решении практических задач. Задания итогового контроля состоят из двух частей: письменного теоретического опроса (от 6 до 12 вопросов) и практических заданий (от 1 до 3), выполняемых на компьютере.

Во время проведения итогового контроля (зачета или экзамена) студентам не разрешается пользоваться вспомогательными материалами Их использование, а также попытки общения с другими студентами или иными лицами, в т.ч. с применением электронных средств связи, перемещения без разрешения экзаменатора и т.д., являются основанием для удаления студента из аудитории с последующим выставлением в ведомость неудовлетворительной оценки.

IV. КОНСПЕКТЫ ЛЕКЦИЙ

Тема 1. Информатика как часть общечеловеческой культуры. Информация

Информатика – это наука, изучающая все аспекты получения, хранения, переработки, передачи и использования информации.

Слово «Информатика» образовалось из 2-х французских слов INFORmation (информация) autoMATIQUE (автоматика). Первоначально в научный обиход оно вошло во Франции в 60-е годы для обозначения автоматической переработки информации в широкой области научной и производственной деятельности человека. В качестве источников информатики обычно называют документалистику (изучение рациональных средств и методов повышения эффективности документооборота) и кибернетику.

Задачи информатики

1. Исследование информационных процессов любой природы.
2. Разработка информационной техники и создание новейшей технологии переработки информации на базе полученных результатов исследования информационных процессов.
3. Решение научных и инженерных проблем создания, внедрения и обеспечения эффективного использования компьютерной техники и технологии во всех сферах общественной жизни.

Основные направления информатики: теоретическая информатика, кибернетика, программирование, информационные системы, вычислительная техника, информатика в обществе, информатика в природе.

Этапы развития информатики: ручной, механический, электромеханический, электронный.

Классификация ЭВМ:

1. По истории создания*:*

I поколение: элементарная база — лампы, оперативная память на электронно-лучевых трубках и ферритовых сердечниках, быстродействие до 20000 оп/сек., охлаждение, однопрограммность.

II поколение: элементарная база — полупроводниковые транзисторы, быстродействие 10^4–10^5 оп/сек. Объем памяти — до 150 слов при длине слова до 50 двоичных разрядов. Программирование велось на алгоритмических языках Фортран, Алгол, Кобол.

III поколение: элементарная база — интегральные схемы (ИС), быстродействие 10^6–10^7 оп/сек. Резко снижены габариты и энергопотребление ЭВМ. Оперативная память строилась на ИС и достигала объема 10^5–10^6 байт. Унифицировались периферийные устройства. Появился широкий выбор языков программирования.

IV поколение: элементарная база — большие и сверхбольшие ИС (БИС и СБИС). Быстродействие 10^7–10^8 оп/сек. Формируются два направления — многопроцессорные и персональные ЭВМ.

2. По размерам*:* суперЭВМ, большие ЭВМ, малые ЭВМ, микроЭВМ

Классификация РС:

1. Стационарные.
2. Переносные: портативные, блокноты, электронные словари, органайзеры (электронные записные книжки).

Информационное общество – это общество, в котором большинство работающих занято производством, хранением, переработкой или реализацией информации, особенно высшей ее формой – знаний.

В период перехода к информационному обществу необходимо подготовить человека к быстрому восприятию и обработке больших объемов информации, овладению им современными средствами, методами и технологией работы. Поэтому современный человек должен иметь определенный уровень информационной культуры.

Информационное общество характеризуется – 50% людей занято в сфере информационных услуг.

Информационная система – взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Информационная технология – процесс, использующий совокупность средств и методов отбора, обработки и передачи данных для получения информации нового качества.

Сигнал – физический процесс, имеющий информационное значение, установленное принятым соглашением. В обычной жизни под сигналом понимают физический процесс, который человек воспринимает как звук, свет и т. д. Данные – зарегистрированные сигналы. Данные могут рассматриваться как признаки или записанные наблюдения, которые по каким-то причинам не используются, только хранятся. Если появляется возможность использовать эти данные для уменьшения неопределенности в чем-либо, то данные превращаются в информацию.

Информация – сведения об объектах и явлениях окружающей среды, их параметрах, свойствах и состоянии, которые уменьшают имеющуюся о них степень неопределенности, неполноты знаний.

Одной из важнейших разновидностей информации является информация экономическая. Экономическая информация – совокупность сведений, отражающих социально-экономические процессы и служащих для управления этими процессами и коллективами людей в производственной и непроизводственной сфере.

Свойства информации: объективность, достаточность (полнота), достоверность, доступность, актуальность, адекватность, своевременность, полнота, энтропия информации.

Классификация информации по разным признакам:

1. По месту возникновения: входная, выходная, внутренняя, внешняя.
2. По стабильности: переменная, постоянная.
3. По стадии обработки: первичная, вторичная, промежуточная, результатная.
4. По способу отображения: текстовая, графическая.

Различают две формы представления информации — непрерывную (аналоговую) и прерывистую (цифровую, дискретную).

Непрерывная форма характеризует процесс, который не имеет перерывов и

теоретически может изменяться в любой момент времени и на любую величину (например, речь человека, музыкальное произведение). Цифровой сигнал может изменяться лишь в определенные моменты времени и принимать лишь заранее обусловленные значения (например, только значения напряжений 0 и 3,5 В). Моменты возможного изменения уровня цифрового сигнала задает тактовый генератор конкретного цифрового устройства.

Для преобразования аналогового сигнала в цифровой сигнал требуется провести дискретизацию непрерывного сигнала во времени, квантование по уровню, а затем кодирование отобранных значений.

Дискретизация – замена непрерывного (аналогового) сигнала последовательностью отдельных во времени отсчетов этого сигнала.

Количественное описание информации базируется на вероятностном подходе. За единицу информации принимается один бит. Это такое количество информации, получаем в результате реализации одного из двух равновероятных событий, например, при бросании монеты. Термин «бит» произошел от выражения binary digit, что означает «двоичная цифра», т.е. принимающая значение 0 или 1.

Более крупная единица информации — байт — равна 8 бит. Проверка присутствия или отсутствия на лекции 24 студентов дает лектору три байта информации. Еще более крупная единица информации — 1 Кбайт — равна 1024 байтам. Далее— 1 Мбайт равен 1024 Кбайтам, 1 Гбайт равен 1024 Мбайтам, а 1 Тбайт равен 1024 Гбайтам.

Тема 2. Аппаратное обеспечение ПК

Современный компьютер состоит из двух взаимосвязанных частей – аппаратного (Hardware) и программного (Software) обеспечения. К аппаратному обеспечению относится все то, что можно потрогать руками (микросхемы, диски, платы).

Архитектура компьютера определяется совокупностью ее свойств, существенных для пользователя. Основное внимание при этом уделяется структуре и функциональным возможностям машины, которые можно разделить на основные и дополнительные.

Персональный компьютер — это настольная или переносная ЭВМ, удовлетворяющая требованиям общедоступности и универсальности применения.

Основные блоки ПК и их назначение

Персональный компьютер — универсальная техническая система. Существует понятие базовой конфигурации, которую считают типовой. Понятие базовой конфигурации может меняться. В настоящее время в базовой конфигурации рассматривают четыре устройства: системный блок, монитор, клавиатуру, мышь.

Системный блок представляет собой основной узел, внутри которого установлены наиболее важные компоненты. Устройства, находящиеся внутри системного блока, называют внутренними, а устройства, подключаемые к нему снаружи, называют внешними. Внешние дополнительные устройства, предназначенные для ввода, вывода и длительного хранения данных, также называют периферийными.

Системный блок обычно включает в себя системную плату, блок питания,

накопители на дисках, разъемы для дополнительных устройств и платы расширения с контроллерами – адаптеры внешних устройств.

На системной плате (Mother Board) размещаются самые главные детали:

1. Микропроцессор (МП) - это центральный блок ПК, предназначенный для управления работой всех блоков машины. В его состав входят:

– устройство управления (УУ) — формирует и подает во все блоки машины в нужные моменты времени определенные сигналы управления (управляющие импульсы); формирует адреса ячеек памяти, используемых выполняемой операцией, и передает эти адреса в соответствующие блоки ЭВМ; опорную последовательность импульсов устройство управления получает от генератора тактовых импульсов. В УУ входят регистр команд – запоминающий регистр, в котором хранится код команды, дешифратор операций и т.д.;

– арифметико-логическое устройство (АЛУ) - предназначено для выполнения всех арифметических и логических операций над числовой и символьной информацией;

– математический сопроцессор - используется для ускоренного выполнения операций числами, для вычисления некоторых трансцендентных, в том числе тригонометрических, функций. Математический сопроцессор имеет свою систему команд и работает параллельно (совмещение во времени) с основным МП, но под управлением последнего;

– микропроцессорная память (МПП) - служит для кратковременного хранения, записи и выдачи информации, непосредственно используемой в вычислениях в ближайшие такты работы машины;

– регистры - быстродействующие ячейки памяти различной длины;

– интерфейсная система микропроцессора - реализует сопряжение и связь с другими устройствами ПК;

2. Генератор тактовых импульсов - он генерирует последовательность электрических импульсов; частота генерируемых импульсов определяет тактовую частоту машины.

Основными характеристиками процессоров являются разрядность и быстродействие. Разрядность — это число одновременно обрабатываемых бит. Быстродействие — число выполняемых команд в секунду. Быстродействие связано с тактовой частотой, на которой работает процессор. Чем выше тактовая частота, тем выше и быстродействие.

3. Источник питания - это блок, содержащий системы автономного и сетевого энергопитания ПК.

4. Таймер - это внутримашинные электронные часы, обеспечивающие при необходимости автоматический съем текущего момента времени (год, месяц, часы, минуты, секунды и доли секунд).

5. Системная шина - это основная интерфейсная система компьютера, обеспечивающая сопряжение и связь всех его устройств между собой. Системная шина обеспечивает три направления передачи информации:

1) между микропроцессором и основной памятью;

2) между микропроцессором и портами ввода-вывода внешних устройств;

3) между основной памятью и портами ввода-вывода внешних устройств.

Запоминающие устройства ПК

Регистровая КЭШ-памятъ — высокоскоростная память сравнительно большой емкости, являющаяся буфером между ОП и МП. Используется для ускорения выполнения операций за счет запоминания на некоторое время полученных ранее данных, которые будут использоваться процессором в ближайшее время..

Основная память делится на оперативное (RAM — Random Access Memory — память вольным доступом) и постоянное (ROM — Read-Only Memory) запоминающие устройство.

Оперативное запоминающее устройство предназначено для хранения информации (программ и данных), непосредственно участвующей в вычислительном процессе на текущем этапе функционирования ПК. ОЗУ — энергозависимая память: при отключении напряжения питания информация, хранящаяся в ней, теряется.

Постоянное запоминающее устройство используется для хранения неизменяемой информации: загрузочных программ операционной системы, программ тестирования устройств компьютера и некоторых драйверов базовой системы ввода-вывода (BIOS — Base Input-Output System) и др. Из ПЗУ можно только считывать информацию, запись информации в ПЗУ выполняется вне ЭВМ в лабораторных условиях. ПЗУ — энергонезависимое запоминающее устройство.

Внешние запоминающие устройства

Внешние запоминающие устройства (ВЗУ) предназначены для долговременного хранения программ и данных. Устройства выполняют в виде накопителей, носителями информации в которых могут служить диски и ленты. Накопители могут быть со сменными носителями и со стационарными носителями информации.

По способу доступа ВЗУ делятся на устройства прямого доступа и устройства последовательного доступа. Накопители на дисках — устройства прямого доступа, так как позволяют обратиться непосредственно к любому месту дискового пространства. Накопители на магнитных лентах — устройства последовательного доступа, так как доступ к нужной информации требует предварительного просмотра всей предыдущей.

Основные характеристики ВЗУ— информационная емкость и время доступа.

Диски относятся к машинным носителям с прямым доступом, это означает, что ПК может «обратиться» к дорожке, начинается участок с искомой информацией или куда нужно записать нужную информацию, где бы ни находилась головка записи/чтения накопителя.

Магнитные диски (МД) относятся к магнитным машинным носителям информации. В качестве запоминающей среды у них используются магнитные материалы со специальными свойствами (с прямоугольной петлей гистерезиса), позволяющими фиксировать два магнитных состояния — два направления намагниченности. Каждому из этих состояний ставятся в соответствие двоичные цифры: 0 и 1. Диски бывают жесткими и гибкими, сменными и встроенными в ПК. Устройство для чтения и записи информации на магнитном диске называется дисководом.

Все диски характеризуются своим диаметром или, иначе, формфактором.

Информация на МД записывается и считывается магнитными головками вдоль концентрических окружностей — дорожек. Количество дорожек на МД и их информационная емкость зависят от типа МД, конструкции накопителя на МД, качества магнитных головок и магнитного покрытия.

Каждая дорожка МД разбита на сектора. В одном секторе дорожки может быть помещено 128, 256, 512 или 1024 байт, но обычно 512 байт данных. Обмен данными между НМД и ОП осуществляется последовательно целым числом секторов. Кластер — это минимальная единица размещения информации на диске, состоящая из одного или нескольких смежных секторов дорожки.

При записи и чтении информации МД вращается вокруг своей оси, а механизм управления магнитной головкой подводит ее к дорожке, выбранной для записи или чтения информации.

Накопители на дисках очень разнообразны. Существуют накопители на гибких магнитных дисках (НГМД), накопители на жестких магнитных дисках (НЖМД), накопители на оптических дисках.

ВНЕШНИЕ УСТРОЙСТВА

Внешние устройства ПК обеспечивают взаимодействие машины с окружающей средой: пользователями, объектами управления и другими ЭВМ. ВУ весьма разнообразны и могут быть классифицированы по ряду признаков.

1. Устройствам ввода информации.

1) Клавиатура — устройство для ручного ввода числовой, текстовой и управляющей информации в ПК. С помощью клавиатуры управляют компьютерной системой.

2) Графические планшеты (диджитайзеры) - для ручного ввода графической информации, изображений путем перемещения по планшету специального указателя (пера).

3) Сканеры (читающие автоматы) — для автоматического считывания с бумажных носителей и ввода в ПК машинописных текстов, графиков, рисунков, чертежей.

4) Цифровые фотокамеры. Как и сканеры, эти устройства воспринимают графические данные с помощью приборов с зарядовой связью, объединенных в прямоугольную матрицу.

5) Манипуляторы (устройства указания) Они предназначены для ввода графической информации на экран дисплея путем управления движением курсора по экрану с последующим кодированием координат курсора и вводом их в ПК:

– джойстик—рычаг;

– мышь, самый распространенный манипулятор.

– трекбол — шар в оправе, в отличие от мыши он устанавливается стационарно, и его шарик приводится в движение ладонью руки.

– световое перо;

– инфракрасная мышь отличается от обычной наличием устройства беспроводной связи с системным блоком и т.д.

2. Устройства вывода информации.

1. Принтеры — печатающие устройства для регистрации информации на бумажный носитель. Они являются наиболее развитой группой ВУ ПК и насчитывают до 1000 разных модификаций.

1) Матричные принтеры.

2) Лазерные принтеры.

3) Струйные принтеры.

2. Графопостроители (плоттеры) — для вывода графической информации (графиков, чертежей, рисунков) из ПК на бумажный носитель.

3. Монитор - это устройство визуального представления данных. Его основными потребительскими параметрами являются: размер, шаг маски экрана, максимальная частота регенерации изображения, класс защиты.

3. Средства связи и телекоммуникации

1. Модем - коммуникационное устройство, позволяющее передавать цифровые данные по аналоговой телефонной линии. Он осуществляет преобразование данных с компьютера в последовательность дискретных (разнотипных) сигналов и их отправку по аналоговой телефонной линии. На другом конце они расшифровываются принимающим модемом путем аналого-цифрового преобразования.

2. Факс-модем - это устройство, сочетающее возможности модема и средства для обмена факсимильными изображениями с другими факс-модемами и обычными телефаксными аппаратами.

4. Устройства речевого ввода-вывода.

Эти устройства относятся к быстроразвивающимся средствам мультимедиа. Устройства речевого ввода — это различные микрофонные акустические системы, "звуковые мыши", например, со сложным программным обеспечением, позволяющим распознавать произносимые человеком буквы и слова, идентифицировать их и закодировать.

Устройства речевого вывода — это различные синтезаторы звука, выполняющие преобразование цифровых кодов в буквы и слова, воспроизводимые через громкоговорители (динамики) или звуковые колонки, подсоединенные к компьютеру.

Средства мультимедиа (multimedia — многосредовость) — это комплект аппаратных и программных средств, позволяющих человеку общаться с компьютером, используя самые разные, естественные для себя среды: звук, видео, графику, тексты, анимацию и др. К средствам мультимедиа относятся устройства речевого ввода и вывода информации; высококачественные видео- (video-) и, звуковые (sound-) платы, платы видеозахвата (videograbber), снимающие изображение с видеомагнитофона или видеокамеры и вводящие его в ПК; высококачественные акустические и видеовоспроизводящие системы с усилителями, звуковыми колонками, большими видеоэкранами, а так же внешние запоминающие устройства большой емкости на оптических дисках, часто используемые для записи звуковой и видеоинформации.

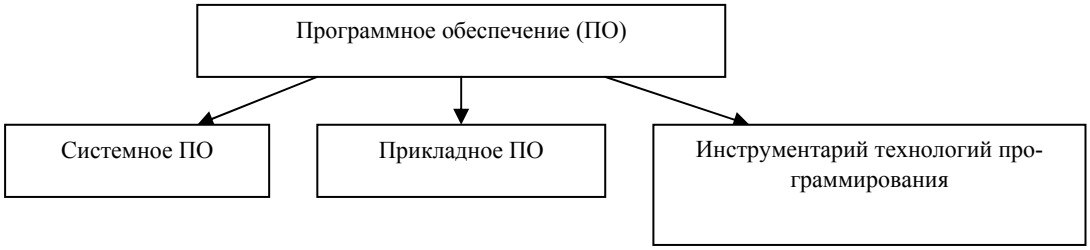
Тема 3. Программное обеспечение ПК

Программа – упорядоченная последовательность команд (инструкций) ком-

пьютера для решения задачи.

Программный продукт - комплекс взаимосвязанных программ для решения определенной задачи массового спроса, подготовленный к реализации как любой вид промышленной продукции.

Интегрированное программное обеспечение — набор нескольких программных продуктов, функционально дополняющих друг друга, поддерживающих единые информационные технологии, реализованные на общей вычислительной и операционной платформе.



Системное программное обеспечение - совокупность программ и программных комплексов для обеспечения работы компьютера и компьютерных сетей.

Базовое программное обеспечение - минимальный набор программных средств, обеспечивающих работу компьютера.

1. Операционная система - совокупность программ, предназначенных для управления ресурсами ЭВМ, организации диалога пользователя с ЭВМ, исполнения программ пользователя.

Основная функция всех операционных систем - посредническая. Она заключается в обеспечении нескольких видов взаимодействия (интерфейса):

- интерфейс пользователя (взаимодействие между пользователем и программно-аппаратными средствами компьютера);
- аппаратно-программный интерфейс (взаимодействие между программным и аппаратным обеспечением);
- программный интерфейс (взаимодействие между разными видами программного обеспечения).

Программы, работающие под управлением операционной системы, называют приложениями.

Сервисное программное обеспечение

Сервисное программное обеспечение (программы-утилиты) — программы и программные комплексы, которые расширяют возможности базового программного обеспечения и организуют более удобную среду работы пользователя.

1. Программы диагностики работоспособности компьютера и обслуживания дисков (служебные программы).

1) Форматирование диска (Format) – разбиение диска на дорожки и сектора.

2) Дефрагментация диска (Disk Defragments) – процедура по переносу информации из одних кластеров в другие, в результате которой доступ к любой информации будет более быстрым.

3) Проверка диска (Scandisk) – программа обнаружения ошибок, связанных со сбоями в процессе записи на диск.

4) Корзина - восстановление удаленных файлов.

5) Очистка диска – программа освобождает место на диске путем удаления временных файлов (в корзине, из Интернета, для быстрого просмотра).

6) Сведения о ресурсах (Sistem Information) – параметры аппаратного обеспечения; программная среда; программа отображает неполадки компьютера, но не устраняет их.

2. Программы архивирования данных.

Архивный файл – это специальным образом организованный файл, содержащий в себе один или несколько файлов в сжатом или несжатом виде и служебную информацию об именах файлов, дате и времени их создания или модификации, размерах и т. п.

Архивация (упаковка) – помещение (загрузка) исходных файлов в архивный файл в сжатом или несжатом виде.

Цели сжатия файлов:

– обеспечение более компактного размещения информации на диске;
– сокращение времени и стоимости передачи информации по каналам связи в сетях;
– упрощение переноса файлов с одного диска на другой;
– защита информации от несанкционированного доступа;
– защита от заражения вирусами.

Степень сжатия информации характеризуется коэффициентом сжатия: $K_c = \frac{V_c}{V_0} \cdot 100\%$, где V_c – объем сжатого файла, V_0 – объем исходного файла.

Хорошо сжимаются графические и текстовые файлы. Слабо сжимаются файлы исполняемых программ.

Разархивация (распаковка) – процесс восстановления файлов из архива точно в таком виде, какой они имели до загрузки в архив. При распаковке файлы извлекаются из архива и помещаются на диск или в оперативную память.

Наиболее распространенные программы – архиваторы: WinRAR и WinZIP.

3. Антивирусные программы.

Компьютерным вирусом называется специально написанная программа, способная самопроизвольно присоединяться к другим программам, создавая свои копии и внедряя их в файлы, системные области компьютера и в вычислительные сети с целью нарушения работы программ, порчи файлов и каталогов, создания всевозможных помех в работе на компьютере.

Прикладное программное обеспечение

1. Текстовые редакторы.

Основные функции текстовых редакторов заключаются в автоматизации ввода и редактирования текстовых данных.

В отличие от текстовых редакторов, текстовые процессоры позволяют не только вводить и редактировать текст, но и форматировать его, т.е. оформлять. Соответственно, к основным средствам текстовых процессоров относятся средства обеспечения взаимодействия текста, графики, таблиц и других объектов.

Назначение настольных издательских систем состоит в автоматизации процесса верстки полиграфических изданий.

2. Графические редакторы.

3. Системы управления базами данных (СУБД).

4. Электронные таблицы.

5. Системы автоматического проектирования.

Программы этого класса предназначены для автоматизации проектно-конструкторских работ, связанных с разработкой чертежей, схем, диаграмм, графическим моделированием и конструированием, созданием библиотеки стандартных элементов чертежей и их многократным использованием, созданием демонстрационных иллюстраций и мультфильмов.

6. Системы автоматизированного управления.

Системы автоматизированного управления - самый представительный класс программных продуктов, включающий в себя программные продукты:

- автоматизированного бухгалтерского учета;

- управления финансовой деятельности;
- управления персоналом (кадровый учет);
- управления материальными запасами;
- управления производством;
- банковские информационные системы и т.п.

7. Офисные приложения.

Данный класс программных продуктов охватывает программы, обеспечивающие организационное управление деятельностью офиса: элементарные текстовые редакторы; органайзеры; автоматизированный перевод документов; средства проверки орфографии и распознавания текста; электронная почта.

8. Экспертные системы.

Предназначены для анализа данных, содержащихся в банках знаний, и выдачи рекомендаций по запросу пользователя. такие системы применяют в тех случаях, когда исходные данные хорошо формализуются, но для принятия решения требуются обширные специальные знания. Характерными областями использования экспертных систем являются юриспруденция, медицина, фармакология, химия.

9. Редакторы HTML (Web- редакторы).

Это особый класс редакторов, объединяющий в себе свойства текстовых и графических редакторов. Они предназначены для создания и редактирования так называемых Web- документов (Web- страниц Интернета). Программы этого класса эффективно используют для подготовки электронных документов и мультимедийных изданий.

10. Геоинформационные системы (ГИС)

ГИС предназначены для автоматизации картографических и геодезических работ на основе информации, полученной топографическими или аэрокосмическими методами.

11. Автоматизация научно-исследовательских работ.

Программы, автоматизирующие научно-исследовательские работы, используют вычислительную мощь компьютера для решения расчетных задач.

12. Программные средства мультимедиа.

Этот класс программных продуктов является относительно новым. Основное назначение программных продуктов мультимедиа - создание и использование аудио- и видеоинформации для расширения информационного пространства пользователя.

13. Системы видеомонтажа.

Системы видеомонтажа предназначены для цифровой обработки видеоматериалов, их монтажа, создания видеоэффектов, устранения дефектов, наложения звука, титров и субтитров.

14. Компьютерная обработка звука. Музыкальные редакторы.

Компьютерная обработка звука ориентирована на систему цифровой записи. Музыкальные редакторы позволяют обрабатывать звук, записанный на звуковой носитель; реставрировать старые записи с помощью встраиваемых приложений; осуществлять нотную запись; создавать многоканальную запись; подготавливать звуковые файлы к публикации в Интернет.

15. Обучающие программы.

Обучающие программы предназначены для самостоятельного изучения данной темы, широко используются в заочном или дистанционном образовании.

Инструментарий технологии программирования

Это совокупность программ и программных комплексов, обеспечивающих технологию разработки, отладки и внедрения создаваемых программных продуктов.

Инструментарий технологии программирования обеспечивает процесс разработки программ и включает специализированные программные продукты, которые являются инструментальными средствами разработчика. Программные продукты данного класса поддерживают все технологические этапы процесса проектирования, программирования (кодирования), отладки и тестирования создаваемых программ. Пользователями технологии программирования являются системные и прикладные программисты.

Инструментарий технологии программирования включают в себя следующие классы:

- средства для создания приложений (языки и системы программирования);
- средства для создания информационных систем (CASE - технологии).

Языки программирования - формальный язык для описания алгоритма решения задачи на компьютере.

Системы программирования - хорошо интегрированная система, включающая в себя как минимум:

- специализированный текстовый редактор;
- компилятор для перевода текста программы в машинный код;
- библиотека функций;
- редактор связей для связывания модулей (файлов с исходными текстами) и стандартных функций, находящихся в библиотеках;
- исполнимый код - это законченная программа с расширением .COM или .EXE, которую можно запустить на любом компьютере, где установлена операционная система, для которой эта программа создавалась;
- справочную систему;
- отладчик, который позволяет анализировать работу программы во время ее выполнения по шагам.

CASE-технологии - программный комплекс, автоматизирующий весь технологический процесс анализа, проектирования, разработки и сопровождения сложных программных систем.

Тема 4. Файловая система.

Norton-подобные операционные оболочки

Файловая система (ФС) является важной частью любой операционной системы, которая отвечает за организацию хранения и доступа к информации на каких-либо носителях.

ФС - это совокупность именованных наборов данных и программ на внешних носителях, структуру и организацию которых поддерживает операционная система.

В широком смысле понятие ФС включает:

- совокупность всех файлов на диске,
- наборы служебных структур данных, используемых для управления файлами, такие как, например, каталоги файлов, дескрипторы файлов, таблицы распределения свободного и занятого пространства на диске,
- комплекс системных программных средств, реализующих управление файлами, в частности операции по созданию, уничтожению, чтению, записи, именованию файлов, установке атрибутов и уровней доступа, поиску и т.д.

Различие между файловыми системами заключается, в основном, в способах распределения пространства между файлами на диске и организации на диске служебных областей.

Файл - это поименованная совокупность элементов информации хранящаяся на магнитных носителях - дисках, лентах.

Имена файлов записываются следующим образом: ИМЯ . ТИП, где ИМЯ набор символов (короткое не более 8 символов, длинное – 256) латинского алфавита, цифр и специальных символов ~ & @ ( ) % { } _ # $, а ТИП или РАСШИРЕНИЕ файла состоит из не более чем 3 символов. В отличие от имени тип может отсутствовать в спецификации файла.

Тип файла используется для классификации, определения принадлежности к какой-то группе с общими свойствами.

При использовании имен файлов в качестве параметров команд ОС необходимо указывать адрес или путь к файлу. Путем к файлу называется цепочка символов, начиная с имени дисковода, корневого каталога и последующих подкаталогов вплоть до каталога, содержащего необходимый файл.

Имя дисковода - это одна из букв латинского алфавита. Персональный компьютер имеет несколько накопителей на магнитных носителях, исходя из этого, принято обозначать A: и B: - гибкие диски, C: D: и т.д. - жесткие диски.

Каталог - это справочник файлов и библиотек со ссылками на их расположение, содержащее информацию о файлах (имя, тип, размеры в байтах, дата и время создания, атрибуты) и других каталогах, называемых подкаталогами, используется операционной системой для определения местоположения файла.

На каждом диске имеется один главный или корневой каталог. Каталоги, входящие в корневой каталог называются подкаталогами 1-го уровня. Каталоги, входящие в состав подкаталога 1-го уровня называются подкаталогами 2-го уровня и т.д. Каждый подкаталог является оглавлением, содержащим перечень имен файлов и подкаталогов, возможны варианты, когда в оглавлении имеются только имена файлов.

Организация файловой системы

Принцип организации файловой системы — табличный. Поверхность жесткого диска рассматривается как трехмерная матрица, измерениями которой являются номера поверхности, цилиндра и сектора. Под цилиндром понимается совокупность всех дорожек, принадлежащих разным поверхностям и находящихся на

равном удалении от оси вращения. Данные о том, в каком месте диска записан тот или иной файл, хранятся в системной области диска в специальных таблицах размещения файлов (FAT-таблицах). К FAT-таблицае предъявляются особые требования надежности, и она существует в двух экземплярах, идентичность которых регулярно контролируется средствами операционной системы.

Наименьшей физической единицей хранения данных является сектор. Размер сектора равен 512 байт. Поскольку размер FAT- таблицы ограничен, то для дисков, размер которых превышает 32 Мбайт, обеспечить адресацию к каждому отдельному сектору не представляется возможным. В связи с этим группы секторов условно объединяются в кластеры. Кластер является наименьшей единицей адресации к данным. Размер кластера, в отличие от размера сектора, не фиксирован и зависит от емкости диска.

Операционные системы MS-DOS, OS/2, Windows 95, Windows NT реализуют 16-разрядные поля в таблицах размещения файлов. Такая ФС называется FAT 16. Она позволяет разместить в FAT-таблицах не более 65 536 записей (2^{16}) о местоположении единиц хранения данных и, соответственно, для дисков объемом от 1 до 2 Гбайт длина кластера составляет 32 Кбайт (64 сектора).

Высокопроизводительная ФС HPFS (High Performance File System) позволяет использовать жесткие диски объемом до 2 Терабайт. Кроме того, она поддерживает разделы диска размером до 512 Гб и позволяет использовать имена файлов длиной до 255 символов. В HPFS по сравнению с FAT уменьшено время доступа к файлам в больших каталогах.

ФС VFAT (Virtual FAT), реализованная в Windows NT 3.5, Windows 95 (DOS 7.0) - это файловая система FAT, включающая поддержку длинных имен файлов. VFAT использует ту же самую схему распределения дискового пространства, что и файловая система FAT.

ФС FAT32 - усовершенствованная версия файловой системы VFAT, поддерживающая жесткие диски объемом до 2 терабайт. Главные отличия от предыдущих версий FAT состоят в следующем. Блок начальной загрузки на разделах с FAT32 был увеличен до 2 секторов и включает в себя резервную копию загрузочного сектора, что позволяет системе быть более устойчивой к возможным сбоям на диске. Объем, занимаемый таблицей размещения файлов, увеличился, поскольку теперь каждая запись в ней занимает 32 байта, и общее число кластеров на разделе FAT32 больше, чем на разделах FAT. Соответственно, выросло и количество зарезервированных секторов.

ФС NTFS (New Technology File System) - наиболее предпочтительная файловая система при работе с ОС Windows NT (Windows 2000 и XP также являются NT системами), поскольку она была специально разработана для данной системы. В NTFS значительно расширены возможности по управлению доступом к отдельным файлам и каталогам, введено большое число атрибутов, реализована отказоустойчивость, средства динамического сжатия файлов. NTFS позволяет использовать имена файлов длиной до 255 символов, при этом она использует тот же алгоритм для генерации короткого имени, что и VFAT. NTFS обладает возможностью самостоятельного восстановления в случае сбоя ОС или оборудования, так что дисковый том остается доступным, а структура каталогов не нарушается. NTFS

позволяет хранить файлы размером до 16 эксабайт (2^{64} байт) и располагает встроенным средством уплотнения файлов в реальном времени.

Операционная система Microsoft DOS.

DOS является основной частью программного обеспечения. DOS означает Disk Operating System (дисковая операционная система), которая необходима для запуска и работы компьютера.

DOS является однопользовательской и однозадачной ОС. Диалог пользователя с DOS осуществляется в форме команд, которые записываются в командной строке, начинающейся с приглашения DOS. Командная строка обычно имеет следующий вид: C:\>_

Составные части DOS

1. Базовая система ввода/вывода (BIOS), находящаяся в постоянной памяти (постоянном запоминающем устройстве, ПЗУ) компьютера.
2. Загрузчик ОС (ядро DOS) - это очень короткая программа, находящаяся в первом секторе каждой дискеты с ОС DOS.
3. Дисковые файлы Io.sys и Msdos.sys.
4. Командный процессор находится в файле Command.com на диске, с которого загружается ОС.
5. Драйверы устройств - это специальные программы, которые дополняют систему ввод/вывода DOS и обеспечивают обслуживание новых или нестандартное использование имеющихся устройств.

Операционные оболочки.

Программа-оболочка – это программа, один из модулей которой, называемый резидентным, постоянно находится в оперативной памяти компьютера.

Оболочки позволяет эффективно работать с файловой системой дисков, а также запускать программы на исполнение.

Тема 5. Операционная система Windows

Основные характеристики и история создания.

Первая графическая многооконная операционная оболочка Windows фирмы Microsoft появилась в 1986 г. После своего возникновения она пережила ряд модификаций, но не все из них были удачными. Однако в 1991 г. вышла версия операционной среды Windows 3.1 и несколько позже сетевой вариант Windows 3.11 For WorkGroups, завоевавшие широкое признание миллионов пользователей. Дальнейший ряд Windows-продуктов продолжили высокопроизводительные многозадачные, полнофункциональные операционные системы с графическим интерфейсом Windows 95, Windows 98, Windows 2000 для компьютеров платформы IBM PC.

Основными характеристическими чертами операционной системы Windows 95 являются:

1. Единый графический пользовательский интерфейс, который составляют рабочий стол, окна, панель задач и другие графические объекты (кнопки, пиктограммы, списки и т.п.).

2. Интегрированная операционная система, ядро которой загружается в момент включения компьютера, активизирует графический интерфейс пользователя и обеспечивает полную совместимость с операционной системой MS-DOS.
3. Объектно-ориентированная система.
4. Программная совместимость обеспечение полной независимости программ от аппаратной части компьютера.
5. Вытесняющая многозадачность - свойство операционной системы самостоятельно в зависимости от внутренней ситуации передавать или забирать управление у того или иного приложения, не позволяющее одному приложению занять все аппаратные ресурсы.
6. 32-разрядная операционная система, поддерживающая 16-разрядные приложения без всякой их модификации.
7. Многопоточность - свойство операционной системы выполнять операции одновременно над потоками нескольких 32-битовых приложений. Поток - это некоторая часть процесса, который может быть выделено процессорное время для одновременного выполнения наряду с другими потоками.
8. Сетевые возможности. Хотя ОС Windows предназначена для управления автономным компьютером, но также содержит все необходимые средства для создания небольшой локальной одноранговой сети: совместное использование ресурсов файлового сервера, принтеров, факс-модемов. Windows 95 имеет средства для интеграции компьютера во всемирную сеть: использование электронной почты и других средств коммуникации.
9. Средства обмена данными между приложениями: буфер обмена (Clipboard), технологии DDE (Dynamic Data Exchange), OLE (Object Linking and Embedding).
10. Интерфейс мультимедиа, включает в себя лазерный проигрыватель (CD-плеер), обеспечивает поддержку видеодисков и видеомагнитофонов и т.п.
11. Поддержка длинных имен файлов и папок (до 255 символов).
12. Использование технологии Plug and Play ("включи и работай") позволяет осуществляет функции распознания новых устройств для их установки и настройки, при этом обеспечивает динамическое изменение конфигурации системы и автоматического уведомления об этом программных приложений.
13. Реализация принципа WYSIWYG -What You See Is What You Get ("что видишь, то и получаешь"). Принцип реализуется при выводе на печать информации, полностью соответствующей изображению на экране.
14. Технология AutoPlay позволяет автоматически озвучивать работу с Windows при установленных средствах мультимедиа.
15. Режим MouseKeys позволяет все действия с мышью выполнять через клавиатуру).

По сравнению с Windows 95, Windows 98 включает средства, позволяющие компьютеру работать быстрее без добавления нового оборудования. В состав Windows 98 входит ряд программ, совместное применение которых повышает производительность компьютера:

- Служебные программы позволяют быстрее выполнять программы, проверять жесткий диск на наличие ошибок и освобождать место на диске, обеспечивать бесперебойную работу системы.
- Проверка диска запускается автоматически после неверного выключения ОС.
- Новый Web-узел ресурсов Microsoft Windows Update автоматизирует процесс обновления драйверов и системных файлов и обеспечивает новейшие возможности технической поддержки.
- Обозреватель Интернета Internet Explorer делает ряд функций доступными с рабочего стола Windows: каналы Web - узлов на рабочем столе, возможности поиска в Интернет, панели обозревателя. Приложение Internet Explorer объединяет рабочий стол с Web, благодаря чему рабочий стол и его папки будут выглядеть и действовать так же, как при работе с Web. Такой рабочий стол называется Active Desktop.

Главными новыми технологическими решениями, реализованных в Windows 2000, являются расширение сетевых возможностей и усовершенствование функций защиту информации в сетях. Windows 2000 оснащена целым рядом свойств управления клиентами и серверами, позволяющих снизить общую стоимость эксплуатации операционной системы.

Основные элементы графического интерфейса Windows.

Задача интерфейса - сделать компьютер доступнее, понятнее для пользователя.

По замыслу разработчиков, общий вид монитора ассоциируется с видом обычного рабочего стола, на котором располагаются документы (окна, подобные листам бумаги).

Рабочий стол (DeskTop) - все пространство экрана в среде Windows с расположенными на нем графическими объектами.

Вдоль одной из границ (чаще в нижней части) рабочего стола находится панель задач. Панель задач содержит:

- кнопку ПУСК - главное системное меню;
- доступ ко всем открытым приложениям. При открытии приложения на панели задач появляется соответствующая открытому окну кнопка. Нажатие этой кнопки позволяет быстро перейти в выбранное окно.
- пиктограммы специальных "фоновых" приложений: часы, индикатор текущей раскладки клавиатуры и др.

Главное системное меню предназначено для быстрого запуска программ, поиска файлов, обеспечения доступа к справке, вызов панели управления для настройки компьютера и др. Меню содержит в себе несколько пунктов, при подведении указателя мыши на одном из пунктов автоматически открывается подменю для выбора нужной операции.

1. Пункт Программа позволяет запускать программы.
2. Пункт Документы содержит ярлыки 15 последних открывавшихся документов.
3. Пункт Избранное характерен для ActiveDesktop и позволяют осуществлять доступ к средствам всемирной сети.

4. Пункт Настройка позволяет настраивать: панель задач, в том числе пункты главного меню; панель управления, доступ к которой также можно получить через папку Мой Компьютер; свойства папки Принтеры и др.
5. Пункт Найти позволяет осуществлять поиск папок и файлов на дисках компьютера, в локальной сети, в Интернет. Поиск можно осуществлять, используя маску файлов, дату создания файла, по типу файла, его размеру, поиск по тексту.
6. Пункт Справка выводит справку по работе с Windows.
7. Пункт Выполнить позволяет запускать программы на исполнение, при этом необходимо указать путь и имя запускаемого файла.
8. Пункт Завершение работы необходим для корректного завершения работы с Windows, перезагрузки компьютера, перезагрузки с выходом в режим MS-DOS.

На рабочем столе размещено несколько графических объектов - ярлыков.

Ярлык - маленький файл-указатель, с помощью которого можно быстро получить доступ к объекту (файлу, каталогу, диску, программе).

Ярлык представлен в виде значка (пиктограммы) и названия. Двойной щелчок по ярлыку открывает объект, с которым он связан.

Основными объектами файловой структуры Windows являются файл, каталог, ярлык. С этими объектами можно проделывать следующие операции: создавать, переименовывать, удалять, копировать, перемещать и другие. В Windows, по аналогии с обычными терминами, принято называть каталог - папкой, файл - документом.

Изначально на Рабочем столе расположены значки «Мой компьютер», «Сетевое окружение», «Корзина» и «Портфель».

Мой компьютер отражает содержание всего компьютера целиком. Папка позволяет просмотреть содержимое находящихся на компьютере дисков, доступ к панели управления, сетевому окружению.

Сетевое Окружение - обеспечивает доступ к ресурсам сети, если к ней подключен компьютер.

В Портфель, как и в обычный дорожный портфель, в котором носят нужные документы, помещают файлы и папки, с которыми пользователь работает на нескольких компьютерах: дома и на работе, на настольном и переносном компьютере в дороге. Портфель содержит базу данных, позволяющую сопоставить несколько вариантов одного документа и выбрать оптимальный.

Корзина - специальная папка, предназначена для временного хранения удаленных файлов, папок, ярлыков. Она позволяет восстановить объекты, удаленные по ошибке. Размер корзины устанавливает пользователь. По мере работы следует регулярно очищать корзину, особенно когда возникают проблемы со свободным дисковым пространством.

Структура окна в Windows.

Окно - прямоугольная область экрана, в которой может отображаться приложение, документ или сообщение.

Окно может быть представлено в виде: свернутое на панель задач (минимизировано), нормальное (окно с обрамление), полноэкранное окно (распахнутое на весь экран).

Любое окно Windows имеет ряд стандартных элементов.

Системное меню содержит команды для изменения размеров окна, его перемещения, минимизации, закрытия.

Заголовок окна является удобным управляющим элементом для перемещения окна. В заголовке окно отображается название приложения и документа (имя открытого файла).

Строка меню содержит пункты для открытия ниспадающего меню.

Контекстное меню (всплывающее меню) появляется при нажатии правой кнопки мыши и содержит активные команды для данного объекта.

Рабочее поле окна представляет собой область для размещения окна документа, и которая будет пустой, пока ни один документ не открыт.

Рамка окна - двойная линия, обрамляющая нормальное окно. Она служит управляющим элементом для изменения размеров окна.

Линейки (полосы) прокрутки окаймляют левую и нижнюю стороны окна и служат для перемещения (прокрутки) документа по вертикали и горизонтали.

Панели инструментов представляют собой линейки командных кнопок, предназначенных для быстрого вызова той или иной команды мышью.

Строка состояния находится у нижнего края окна и содержит информацию о режимах работы приложения.

Виды окон

1. Окно приложения представляют собой интерфейсы работающих приложений. Главным свойством окон приложений является то, что они могут перекрывать друг друга и являются независимыми, т.е. не подчинены никакому другому окну.

2. Окно документа всегда подчинены окнам своих приложений и не выходят за их пределы.

3. Диалоговое окно служит для ввода дополнительных параметров, необходимых для выполнения какой-либо команды.

Активное окно - окно приложения, реагирующее в данный момент на действия пользователя. Заголовок активного окна отличается по цвету и яркости от неактивного.

<u>Тема 6.</u> Введение в программирование

Этапы решения задач

Программированием называют планирование, распределение во времени или исполнение заданий или событий. Компьютерное программирование – планирование последовательности шагов, которую должен выполнить компьютер.

Компьютерная программа состоит из команд (инструкций), сообщающих

компьютеру, что он должен делать и каким образом обрабатывать данные. Задачей большинства программ является распознавание данных, их обработка и вывод на экран в требуемом формате.

Процесс создания программы состоит из двух основных этапов: *решение задачи* и ее *реализация*. Этап решения задачи включает в себя:

- *анализ и спецификацию*, т.е. понимание и определение сути задачи, а также технических и функциональных требований к ее решению;
- *общее решение (алгоритм)* - разработку логической последовательности шагов, приводящих к решению поставленной задачи;
- *проверку*, т.е. подтверждение правильности решения, например, повторением всех его этапов.

Этап реализации содержит:

- *конкретное решение* (*программу*) - перевод алгоритма на язык программирования;
- *тестирование* - запуск программы на компьютере, а затем проверка результата вручную. При обнаружении ошибок проводится анализ программы и алгоритма, нахождения источника ошибок и их исправление.

После выполнения перечисленных этапов начинается этап *сопровождения* включающий:

- *использование* (*эксплуатацию*) программы;
- *поддержку* программы, т.е. изменение в соответствии с требованиями пользователей, а также исправление ошибок, выявленных при ее эксплуатации.

При модифицировании программы, необходимо повторить этапы решения и реализации для тех частей программы, которые были изменены.

Совокупность всех трех этапов – решения, реализации и сопровождения – называется *жизненным циклом программы*.

Процесс программирования начинается с анализа задачи, разработки решения, называемого *алгоритмом*. Компьютер не обладает интеллектом, он не способен самостоятельно решать задачи. Компьютер способен лишь очень быстро и безошибочно повторить процедуру решения, подготовленную программистом. Поэтому понимание и анализ задачи является центральной частью процесса программирования.

Каждая компьютерная программа – это реализация определенного алгоритма. *Алгоритм* – устное или письменное описание логической последовательности действий. Алгоритм отражает последовательность действий, которые могли быть выполнены вручную.

Разработав решение, программист проверяет его, выполняя каждый шаг самостоятельно. Если выясняется, что алгоритм не работает, следует снова анализировать задачу и искать новый алгоритм. После получения правильного алгоритма программист переводит его на язык программирования.

Язык программирования – набор правил, символов и специальных слов, используемых для построения программ.

Любой язык программирования представляет собой сильно упрощенную форму естественного языка (как правило, английского), дополненную математи-

ческими символами. За счет ограничения словаря и строгих грамматических правил язык программирования управляет компьютером. Ограниченность и однозначность конструкций языка программирования вынуждает программистов создавать простые и точные инструкции.

Процесс перевода алгоритма на язык программирования называется *кодированием* (*программированием*). Его результатом является код программы (или просто программа), которая проверяется *запуском* (или *выполнением*) на компьютере. Программа, не выдающая желаемых результатов, нуждается в *отладке*.

Отладка программы поиск причины ее ошибочного поведения и изменение фрагментов программы (или даже самого алгоритма) для устранения этой причины.

Реализацией алгоритма называется его совместное кодирование и тестирование. Если не затрачено достаточное время на обдумывание и разработку алгоритма, отладка и исправление программы может потребовать много дополнительных усилий.

Наряду с решением задачи, реализацией алгоритма и сопровождением программы важной частью процесса программирования является *разработка документации.* Документацией называют текст и комментарии, которые упрощают программу для понимания, использования и изменения.

Языки программирования

Компьютер выполняет инструкции встроенной системы элементарных команд - машинного языка, которой состоит из команд в двоичном представлении.

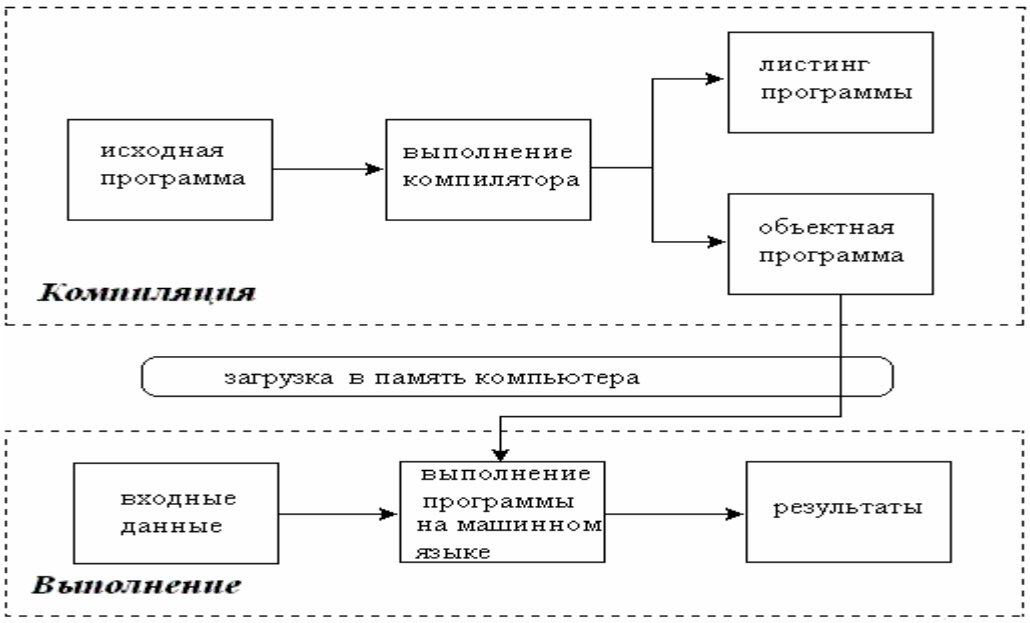
Языки программирования высокого уровня позволяют создавать компьютерные программы на языке, близком к естественному языку. Их легче использовать, чем машинный код. Примерами языков программирования являются Pascal, C++, Fortran, Java и др.

Специальная программа, называемая *компилятором*, переводит программу, написанную на языке высокого уровня в машинный код.

Программа, написанная на языке высокого уровня, называется *исходной программой* и может быть выполнена на любом компьютере в соответствии с используемым компилятором. Это возможно, поскольку большинство языков высокого уровня стандартизировано.

Компилятор транслирует исходную программу в программу на машинном языке, которая называется *объектной программой*. Если программа содержит ошибки, компилятор создает *листинг* программы – текст с сообщениями об ошибках и другой полезной информацией.

Объектная программа отправляется на *выполнение*. Необходимо разделять понятия компиляции и выполнения программы. Во время компиляции запускается компилятор, создающий объектную программу. При выполнении объектная программа загружается в память компьютера, который выполняет команды в соответствии с исходной программой.



Существуют синтаксические (грамматические) и семантические (смысловые) правила создания программы.

Синтаксис – формальный набор правил, определяющий, какие комбинации цифр и букв могут использоваться в языке программирования. Синтаксис однозначно определяет правила построения конструкций языка. При наличии синтаксических ошибок программа не будет скомпилирована.

Семантика определяет значения команд языка.

Тема 7. Базовые конструкции языка C++

Среди современных языков программирования C++ относят к числу наиболее распространенных. В его основу положено значительно меньше синтаксических правил, чем у других языков программирования. Язык менее строго структурирован и предоставляет программисту свободу выбора альтернативных решений одной проблемы.

Характерные для многих языков программирования конструкции языка, напоминающие выражения на английском языке, в C++ встречаются довольно редко. Язык C++ содержит операторы необычного вида и часто использует указатели.

C++ универсален, однако его применение наиболее эффективно в задачах системного программирования – разработке трансляторов, интерфейсов, операционных систем.

Язык C++ поддерживает полный набор структурного и объектно-ориентированного программирования: модульность, блочную структуру программ, раздельную компиляцию, характернее для языков высокого уровня. С другой стороны, он имеет ряд низкоуровневых черт (в частности, операции над битами).

Непосредственным предшественником языка C++ был язык C с классами. В основу языка C было положено значительно меньше синтаксических правил, чем у других языков программирования. В результате для эффективной работы ком-

пилятора языка достаточно всего 256 Кб оперативной памяти. Первоначально, в том виде, в каком его создал Деннис Ритчи в 1972, С содержал всего 27 ключевых слов. После чего язык стал бурно развиваться. В 1983 г., при Американском институте национальных стандартов (ANSI) был создан специальный комитет с целью стандартизации языка. Стандарт ANSI языка С включал уже более 50 ключевых слов.

К настоящему времени в программировании сформировано несколько направлений:

- процедурное программирование;
- структурное (модульное) программирование;
- объектно-ориентированное программирование.

Язык С++ поддерживает все эти направления.

В *процедурном* программировании основное внимание уделяется алгоритму, а именно его эффективности и компактности. Методы процедурного программирования особенно были важны, когда компьютеры не обладали достаточными быстродействием и объемом памяти. Но нельзя сказать, что они утратили свою актуальность сегодня.

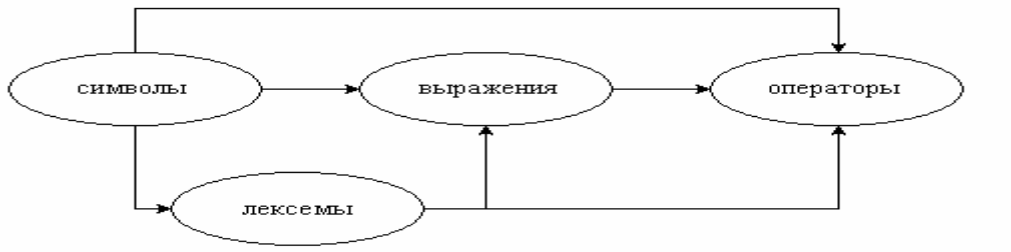
В *структурном* программировании основное внимание уделяется организации данных. Программы делятся на модули таким образом, чтобы данные внутри модулей были скрыты. Применение методов структурного программирования позволяет создавать сложные программные продукты коллективам программистов. Поскольку каждый модуль может быть разработан, скомпилирован и отлажен отдельно. Методы структурного программирования позволяют создавать программы, удовлетворяющие критерию надежности и простые в сопровождении.

В действительности перечисленные направления не исключают, а дополняют друг друга.

Объектно-ориентированное программирование в большей степени, чем структурное программирование, предоставляет возможность создавать программы, обладающие структурированностью, модульностью и абстракциями данных, является современным методом создания сложных программ, в которых недостаточно использования методов структурного программирования.

Состав языка

Любой естественный язык содержит четыре основных элемента: символы, слова, словосочетания, предложения. Алгоритмический язык также включает символы, на основании которых строятся элементарные конструкции (*лексемы*). Из лексем и символов строятся *выражения*, которые в свою очередь образуют *операторы*.



Алфавит языка включает в себя основные неделимые знаки, с помощью которых пишутся все тексты программ. Лексема является минимальной единицей языка, имеющей самостоятельный смысл. *Выражение* задает правило вычисления некоторого значения. *Оператор* представляет собой законченное описание некоторого действия. Любое выражение, заканчивающееся точкой с запятой, является оператором, выполнение которого заключается в вычислении выражения.

Операторы бывают исполняемые и неисполняемые. Первые задают действия над данными, а вторые служат для описания данных и называются операторами описания или просто описаниями.

Для описания сложного действия требуется последовательность операторов. Операторы могут объединяться в сложный оператор или блок.

Объединенная единым алгоритмом совокупность описаний и операторов образуют программу на алгоритмическом языке.

Алфавит языка

Алфавит C++ включает в себя:

- прописные и строчные буквы латинские буквы и знак подчеркивания;
- арабские цифры от 0 до 9;
- специальные знаки: “, { } , | [ ] ( ) + - / % * . \ ‘ ^ ? < = > ! & # ~ $;
- пробельные символы: пробел, табуляция, символы перехода на новую строку.

Из символов алфавита формируются *лексемы* языка. Лексемы делятся на идентификаторы, ключевые (зарезервированные) слова, знаки операций, константы, разделители.

Переменные, идентификаторы

Идентификаторы используются в C++ для именования различных объектов. *Идентификатор* – имя, связанное с данными или функцией программы, которое используется для обращения к этому объекту или функции.

Идентификатор представляет собой последовательность символов произвольной длины, содержащую буквы, цифры и символ подчеркивания, но начинающуюся обязательно с буквы или символа подчеркивания. Прописные и строчные буквы различаются. Пробелы внутри имен не допускаются.

Длина идентификатора по стандарту не ограничена, но некоторые компиляторы и компоновщики накладывают ограничения, например, распознают только первые 31 символ. В именах нельзя использовать термины, являющиеся частью языка C++.

Ключевые слова – зарезервированные идентификаторы, которые имеют специальное значение для компилятора. Их можно использовать только в том смысле, в котором они определены. Язык C++ содержит 63 ключевых слова.

Переменная – именованная область памяти, в которой хранятся данные определенного типа. У переменной есть имя (идентификатор) и значение. Имя служит для обращения к области памяти, в которой хранится переменная. Значение

переменной может изменяться во время выполнения программы. Прежде чем использовать переменную, ее необходимо определить.

Данные, необходимые для работы программы, хранятся в памяти компьютера. Каждая область памяти имеет однозначно определенный адрес, на который ссылаются, когда необходимо сохранить или прочесть данные.

Идентификаторы используются для обозначения определенной области памяти, а компилятор транслирует имена в соответствующие адреса.

Имя переменной должно отражать смысл хранимой величины, быть легко распознаваемым и не содержать символов, которые можно перепутать друг с другом.

Каждый элемент данных должен принадлежать к определенному типу. Тип данных определяет, в каком виде они представлены в компьютере, а также какие преобразования компьютер может к ним применять.

Типы данных

Тип данных – множество допустимых значений данных и набор операций, применимых к этим значениям.

В С++ определены наиболее часто используемые типы данных. Кроме того, программист может сам определять новые типы.

Для описания стандартных (встроенных) типов в С++ используется набор ключевых слов: int, short, long, signed, unsigned, char, flout, double.

Первые шесть используются для представления целых данных разной длины (по количеству занимаемых битов в памяти компьютера). Они могут появляться в программе по отдельности или в некоторых сочетаниях.

int обозначает основной целый тип, которому соответствует стандартная длина слова, принятая на используемой машине. (На IBM – 16 битов). Диапазон значений, как правило, зависит от системы. Для многих персональных компьютеров значение типа int меняется от -32768 до +32767.

long или long int может содержать целое значение, не меньшее максимальной величины, допускаемой типом int, или даже больше чем short или short int : максимальное целое число short не больше чем максимальное число типа int, а может и меньше. Обычно числа типа long бывают больше типа short, а тип int реализуется как один из указанных типов, все зависит от конкретной системы. (В IBM для short отводится 16 бит, а для long - 32 бита).

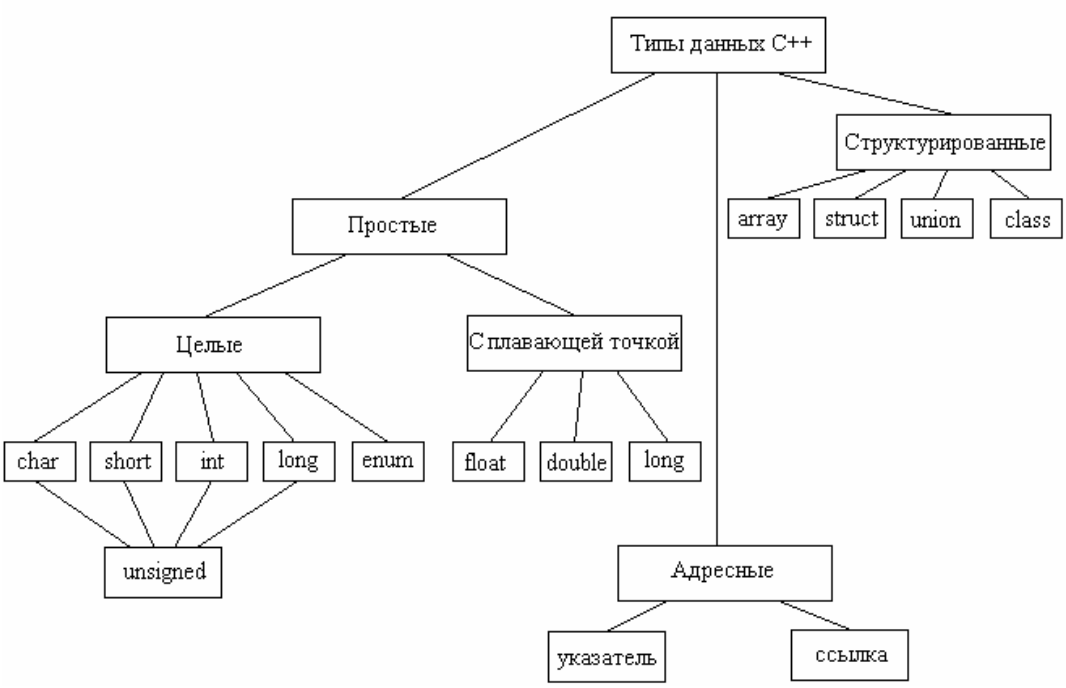
Все эти типы имеют 2 формы: знаковую (signed) и незнаковую (unsigned).

Целые незнаковые константы записываются также как и обычные целые, с тем исключением, что использование знака запрещено. Просто unsigned соответствует unsigned int.

char – самое короткое целое. Значения символьного типа занимают только 1 байт. Наиболее часто этот тип применяется для описания данных, состоящих из отдельного алфавитно-цифрового символа. Их называют символьные переменные. Например, ‘a’, ‘1’, ’+’, ’?’,’з’.

flout и double – числа с плавающей запятой или вещественные, которые могут принимать как положительные так и отрицательные значения. Такие числа имеют целую и дробную части, разделенные точкой. Например, 7.9, 3490.725.

Встроенные типы данных языка C++ подразделят на простые, структурированные и адресные



Объявить переменную означает задать ее имя и тип. Объявление сообщает компилятору, что данный идентификатор связывается с областью памяти, содержимое которого имеет определенный тип.

Синтаксис объявления переменной следующий:

<имя типа> <идентификатор переменной>;

Например,

```
int k;
```

Объявлена переменная с именем k целого типа. Объявление всегда заканчивается точкой с запятой. Таким образом, переменная k может содержать только целое значение. Если компилятор C++ встретит оператор, в котором переменной k будет присваиваться вещественное значение, то он произведет дополнительные действия для преобразования вещественного типа в целое.

Существует возможность объявить сразу несколько переменных одного типа в одном выражении. Для этого имена переменных перечисляются через запятую. Например,

```
int Number, Count;
float cost1, cost2;
unsigned positive;
```

Определяя переменную можно задать также и ее начальное значение, то есть инициализировать переменную. Инициализатор можно записывать в двух

формах – со знаком равенства (=) или в круглых скобках.
Пример:

```
int d, c=5, r=4;
short b(8);
int k, l(145),  m;
unsigned s=0.5;
```

Таким образом можно собрать в один оператор описания переменные одного и того же типа, или наоборот, разбить одно описание на несколько операторов – эффект будет одинаков.

При инициализации символьных переменных их значение требуется заключать в апострофы.

```
char ch='z',  f;
```

Операции и выражения

Знак операции это один или более символов, определяющих действие над операндами. Операции делятся на унарные, бинарные и тернарную по количеству участвующих в них операндов.

Значение переменной можно изменить с помощью операции присваивания =. Например,

```
int summa = 0;
summa = 5;
```

В отличие от алгебраического уравнения в операторе присваивания сначала вычисляется выражение в правой части оператора, а затем оно присваивается отдельной переменной, стоящей слева от знака равенства. Например,

```
int k = 5, m = 1;
m = k;
```

Это означает, что значение переменной m стало равно 5, а не то, что m равно k. Кроме того, выражения m = k и k = m обозначают различные действия. В первом случае обе переменные станут равными пяти, а во втором – единице.

В C++ допускается использование нескольких присваиваний в одном выражении:

```
a = в = c = 0;
```

В любой программе требуется производить некоторые вычисления. Для вычисления значений используются выражения, которые состоят из операндов, знаков операций и скобок. Операнды задают данные для вычислений. Операции задают действия, которые необходимо выполнить.

Основные арифметические операции языка C++ обозначаются стандартными математическими операциями: сложение +, вычитание −, умножение *, деление /. Эти операции, как и операция присваивания, являются бинарными, так как для каждой из них требуется по два аргумента.

Унарный минус используется для определения отрицательных значений, унарный плюс практически не используется, потому что число без знака считается положительным по определению.

Особое внимание требует деление целых чисел. Необходимо помнить, что при делении целых чисел результат операции также целочисленный. Дробная часть просто отбрасывается. Для получения остатка от деления используют одноименную операцию %.

Так как 8 : 2 равно 4, то, 8 / 2 равно 4, 8 % 2 равно 0; 7 : 2 равно 3 (и 1 в остатке), поэтому 7 / 2 равно 3, а 7 % 2 равно 1.

При делении вещественных чисел получается вещественный результат.

Поскольку выражения могут содержать и переменные, допустимо использовать операции присваивания в следующем виде:

```
a = c + 5;
e = a / 2 - 11;
```

После каждого оператора обязательно ставится точка с запятой. Одна и та же переменная может встречаться с обеих сторон знака присваивания.

```
n=n+ 7;
```

Это означает, что сначала складываются значение, содержащееся в переменной с именем n и семь, а затем полученное значение помещается в переменную n, тем самым, заменяя ее предыдущее значение.

Кроме стандартных арифметических операций в С++ введен ряд специальных операций. Из них наиболее часто используемыми являются операция инкремента (увеличение на единицу) ++, и операция декремента (уменьшение на единицу) - - . Использование операция инкремента и дала название языку – С++.

Эти операции унарные (операции с одной переменной). Они могут использоваться с целым и вещественным аргументом. Использование оператора

```
j++ ;
```

эквивалентно оператору

```
j=j+1;
```

А соответственно j- - эквивалентно j = j – 1.

У этих операций существует особенность: они имеют две формы: префиксную, когда ее можно поместить перед переменной и постфиксную – после переменной.

Записанные в таком виде j+ +; + + j; эти операторы эквивалентны – каждый из них увеличивает значение j на единицу. Поэтому в данном случае выбор одной из форм оператора является делом вкуса. Однако С++ позволяет их использование в середине сложных выражений, и тогда их использование может привести к различным результатам.

Примеры:

```
int bar = 1;
cout << ++ bar;
int bar = 1;
cout << bar ++;
```

В первом случае ++bar увеличивает значение bar на единицу, а затем записывает это значение собственно в bar, то bar будет равен 2, и значение 2 будет выведено на экране.

bar ++ определяет значение bar, а потом выполняет приращение, следовательно, устанавливает bar 2, но выводит на экран 1, поскольку вывод происходит

перед выполнением приращение.

В С++ определены операции составного присваивания. Они используются для сокращения записи операторов, содержащих в себе присваивание и арифметическую операцию. Оператор вида <операнд1> += <операнд2> эквивалентен записи <операнд1> = <операнд1> + <операнд2>. Существует составное присваивание со сложением, вычитанием, делением, умножением, с остатком от деления:

+ = ; - = ; * = , / =, % =.

Пример

```
foo + = 3;      // эквивалентно
foo = foo + 3;
```

С++ содержит операции отношения: больше > , больше или равно > = , меньше < , меньше или равно< = , равно = = , не равно! = , не !.

Также определены логические операции: И && и ИЛИ ||. Результатом логической операции является true (истина) или false (ложь). Результатом операции логическое И является значение true, если оба операнда имеют значение true. Результат логического ИЛИ true, если хотя бы один из операндов имеет значение true. Логические операции выполняются слева направо. Например, результат выражения (k>=1)&&(k<10) будет true, если k=5. И это же выражение есть false, если k=0.

Операция определения размера sizeof( ) предназначена для вычисления размера объекта или типа в байтах, и имеет 2 формы: sizeof (выражение) или sizeof(тип). Так результат выполнения sizeof(int) равен 2, так как для хранения данных этого типа в памяти выделяется 2 байта.

С++ имеет одну тернарную операцию. Это условная операция (? :). Ее формат:

<условие> ? <выражение1> : <выражение2>;

Данный оператор позволяет создавать простые условные однострочные выражения, в которых выполняется одно из двух действий в зависимости от значения условия. Данный оператор можно использовать вместо инструкции if/else. Рассмотрим пример, в котором определяется модуль числа с помощью условного оператора

```
fvalue = (fvalue>=0.0) ? fvalue : -fvalue;
```

Рассмотрим другой пример применения условной операции. Требуется, чтобы некоторая величина увеличилась на 1, если она не превышает n, иначе принимала бы значение 1:

```
y = (y < n) ? y+1: 1;
```

Операции выполняются в соответствии с приоритетами. Для изменения порядка выполнения операций используются круглые скобки.

Операции	***Приоритет операций***
++ – – ! sizeof	наивысший приоритет
* / %	
+ –	

< <= > >=
= = !=
&&
||
?:
= += −= наименьший приоритет

Рисунок 4. Приоритеты операций

Выражения состоят из операндов, знаков операций и скобок и используются для вычисления некоторого значения определенного типа. Каждый операнд является, в свою очередь, выражением или одним из его частных случаев – константой или переменной.

Примеры выражений:

(d + 8) / 67

x && y || !z

(t + 5*k) / (f – 56) + 470

Если в одном выражении записано несколько операций одинакового приоритета, унарные операции, условная операция и операция присваивания выполняются *справа налево*, остальные – *слева направо*. Например, a=c=y означает a=(c=y), а a+b+c означает (a+b)+c.

Ввод – вывод на экран. Введение в потоки ввода – вывода

Оператор cout (си-аут) позволяет осуществлять вывод данных на экран монитора. Переменная cout зарезервирована для обозначения выходного потока.

Для того чтобы послать значение на си-аут применяют последовательность cout<< (оператор «направить в» или оператор вставки).

Если необходимо напечатать строку символов, требуется взять ее в кавычки. Можно также напечатать несколько значений одновременно, разделяя их оператором вставки.

cout << “My name is” <<”Tatyana”;

При печати цифр их можно не помещать в кавычки. Возможно объединение текста и цифр.

cout << “ мой адрес: Институтская“ << 26;

В процессе печати можно использовать довольно большое количество специальных символов. Вот некоторые из них:

\n Начало новой строки
\t табулятор
\b возврат назад на один пробел
\f начало новой строки страницы
\\ печать символа обратный слэш
\’ печать символа’
\” печать символа ’’

Пример:

cout << “ He said:\ ” Hello:\” \ n”;

Оператор вставки использует два аргумента. Аргумент слева от << является потоковым выражением (потоковой переменной). Правый аргумент представляет собой строку или выражение, результат которого имеет простой тип. Оператор вставки преобразует правый операнд в последовательность символов и добавляет их выходной поток. Например,

```
cout<<"Результат равен  " << 5*n + 90;
```

Если n равно 10, то на экране появится:

Результат равен 140

Для перехода на новую строку используют манипулятор endl, который также очищает буфер потока. Например,

```
Int x=17, y=21;
cout << "x= "<<x<<endl<<"y="<<y;
```

На экране появится

x=17

y=21

Стандартная библиотека C++ предоставляет пользователю большое количество манипуляторов, позволяющих форматировать ввод-вывод. Манипулятор setw (сокращение от «set width» – «установить ширину») позволяет управлять количеством позиций для вывода следующего за манипулятором элемента данных. Применяется только для форматирования чисел и строк, но не данных типа char. Параметр данного манипулятора – целое выражение, определенное число знаковых позиций для вывода очередного элемента. Данные при выводе выравниваются по правому краю, а свободные позиции слева заполняются пробелами. Например,

```
int ans=33, num=7132;
cout<<setw(4)<<ans<<setw(5)<<num;
```

выведет на экран □□33□7132

```
cout<<setw(1)<<ans<<setw(6)<<num;
```

33□□7132 - поле автоматически расширяется, чтобы вместить двузначное число.

Установка ширины поля является одноразовым действием и влияет только на ближайший элемент вывода.

Контроль числа десятичных позиций при выводе решается с помощью манипулятора setprecision, указывающего количество знаков после запятой. Например,

```
int x=4.856;
cout<<setw(6)<<setprecision(2)<<x;
```

вывод □□□4.85

Для ввода с клавиатуры используют символ cin (си-ин или син) и >> оператор ,,взять из“

```
cin >> Number;
```

Стандартные функции ввода-вывода языка C, также доступны и в C++. Наиболее часто используемыми функциями ввода-вывода языка C, являются printf( ) и scanf( ), которые обеспечивают форматный ввод-вывод и являются достаточно универсальными, особенно при работе с числами, но из-за обилия всевозможных спецификаторов форматирования, становятся иногда громоздкими и трудно чи-

таемыми. Операторы << и >> благодаря понятию перегрузки операторов поддерживают все стандартные типы языка С++, включая классы.

Директива #include

Язык С++ содержит очень небольшое число встроенных функций, но он легко расширяется дополнительными библиотеками.

Операторы cin и cout также являются частью библиотеки. Для их использования необходимо включить в программу соответствующие заголовочные файлы. Это делается с помощью команды #include.

Любая команда, начинающая с решетки называется директивой препроцессора. Она не является выражением языка С++ (поэтому не заканчивается точкой с запятой). Директивы препроцессора могут состоять только из одной строки.

Препроцессор – это программа, действующая как фильтр на этапе компиляции. Так препроцессорная директива #include приказывает компилятору загрузить включаемый файл.

Описания операторов cin и cout находятся в файле с именем iostream.h (input output stream – поток ввода-вывода, h - стандартное расширение заголовочного файла (сокращение от header file).

Синтаксис: имя файла помещается в угловые скобки < >, что указывает препроцессору, что этот файл ищется в стандартном каталоге подключаемых файлов:

```
# include < iostream.h>
```

Компилятор знает, где искать стандартные заголовочные файлы. Если же требуется загрузить заголовочный файл, созданный пользователем, его имя необходимо заключить в кавычки.

```
# include “ myfile.h”
```

Можно также указать полный путь

```
# include “ c \ My \ myfile.h “
```

Для использования манипуляторов при выводе данных также необходимо подключить библиотеку – iomanip.h:

```
# include <iomanip.h>
```

Построение программы

При создании алгоритма решения сложной задачи возможно разбиение решения на решение более простых подзадач. Подпрограммы позволяют сначала записать части программы по отдельности, а затем собрать их в единое работоспособное целое. В языке С++ подпрограммы называются *функциями*, а программа представляет собой набор из одной или более функций. Каждая функция выполняет определенное действие, а все вместе они решают задачу в целом.

Итак, любая программа, написанная на языке С++ есть последовательность выполнения функций, причем одна из них обязательно должна называться main( ). Выполнение программы всегда начинается с выполнения ряда операторов этой функции. Когда main () хочет, чтобы другая функция выполнила свое задание, она вызывает (активизирует) необходимую функцию.

Все функции в языке С++ равноправны: каждая из них (даже main( )) может быть любой другой функцией. Функция может вызывать саму себя (явление рекурсии). Компилятор не ограничивает число рекурсивных вызовов, но операционная система может наложить чисто практические ограничения.

Описание функции состоит из заголовка и тела, и в общем случае, выглядит следующим образом:

Директива препроцессора #include <iostream.h>

Заголовок <тип результата> имя функции (список аргументов)

Тело функции {

Описание *пять типов операторов*

Присваивание

Функция

Управление

Пустой оператор

}

Предполагается, что любая часть описание функции может не использоваться, кроме имени функции. Пример простейшей программы:

```
# include < iostream.h>
main ( )
{
 cout << “HeLLo!!! ”;
}
```

Данная программа просто выведет на экран монитора HeLLo!!!

Фигурные скобки отмечают начало и конец тела функции. В круглых скобках в общем случае содержится информация, передаваемая этой функции.

Если тип результата не указывается, то предполагается, что функция возвращает значение типа int. Если функция не возвращает результат, указывается слово void. Некоторые компиляторы требуют обязательного указания типа возвращаемого результата перед именем функции.

Наличие списка аргументов и описаний аргументов также не является обязательным. Но круглые скобки всегда должны присутствовать после имени функции. Аргументы функции – это величины, которые необходимо передать из вызывающей функции в вызываемую функцию.

Часто употребляется описание типов аргументов сразу при их объявлении внутри круглых скобок. Тип аргумента может быть любым. Если аргументов несколько, их описания (тип плюс имя) разделяются запятыми. Если функции не передаются величины, то вместо списка аргументов необходимо задавать ключевое слово void. Локальные переменные отличные от аргументов, описываются внутри тела функции.

Возвращает значение оператор return, с помощью которого можно передать в вызывающую функцию только одно значение. Возвращаемое значение можно брать в круглые скобки после ключевого слова return. Круглые скобки не являются обязательными.

Пример

Напишем программу, содержащую три функции: непосредственно main и функции, вычисляющие значения куба и квадрата некоторого целого числа.

```
#include <iostream.h>
int square (int x)
        { return x*x; }
int cube (int y)
        { return y*y*y; }
  void main ( )
  {     cout<< “Квадрат числа 15 равен”<<square(15)<<endl;
        cout<< “Куб числа 10 равен”<<square(10)<<endl;
  }
```

В каждой из трех функций левая и правая фигурные скобки указывают на начало и окончание тела функции, т.е. начало и окончание исполняемых выражений. Их называют операторные скобки.

В первой строке расположена директива препроцессора, подключающая заголовочный файл iostream.h, необходимый для работы оператора cout.

Далее следует заголовок и тело функции square. Функция имеет один аргумент целого типа, описание которого расположено в круглых скобках после имени функции. Функция square возвращает целочисленный результат в вызываемую функцию, на это указывает ключевое слово int, стоящее перед именем функции. Возвращение результата из функции осуществляет оператор return. Результатом является выражение после указанного оператора. После оператора обязательно ставится точка с запятой (как требует синтаксис языка C++).

Затем в программе расположено описание функции cube, аналогичное предыдущей функции.

Ниже идет описание функции main. Перед именем функции стоит ключевое слово void, сообщающее компилятору, что функция не возвращает значений. Функция не имеет аргументов – в ее заголовке пустые круглые скобки. В скобках также можно указать void,что также будет указывать на отсутствие аргументов. Выполнение любой программы всегда начинается с первого оператора функции main, имеющего в данной программе вид:

```
cout<< “квадрат числа 15 равен”<<square(15)<<endl;
```

В этом операторе происходит вызов функции square для аргумента 15, следовательно, выполняется оператор данной функции, т.е. 15 умножается на 15 и результат возвращается в main и при помощи cout выводится на экран в виде:

Квадрат числа 15 равен 225

После этого main продолжает выполнение своих операторов, напечатав сообщение

Куб числа 10 равен 100

На этом программа завершает свое выполнение.

Любая функции в C++ может возвращать не более одного значения, т.е. либо не возвращать ничего, либо возвращать единственное значение. Аргументов же у функции может быть сколько угодно, в том числе ни одного.

Пример. Опишем функцию, возвращающую среднее значение трех величин.

```
// определение функции Average
long Average (long val1, long val2, long val3 )
{
long sum = val 1 + val 2 + val 3;
return sum / 3;
}
```

Такая функция может быть вызвана любой другой функцией. Пусть в программе объявлены некоторые переменные: long a, a1, a2, a3; тогда возможен вызов вида

```
a = Average (a1, a2, a3);
```

или

```
a = Average (525, 675, 819);
```

В случаях, когда вызываемая функция не возвращает значение, ее вызов представляет просто имя функции, а в круглых скобках – фактические значения аргументов функции.

Константы

Иногда требуется, чтобы значение переменной оставалось постоянным в течении всего времени работы программы. Такие переменные называются *константными*.

Для их создания необходимо написать определение для переменной с добавлением ключевого слова const перед типом

```
const int Top = 12;
```

Константные переменные используются в программе также как обычные. Единственное отличие заключается в том, что начальные значения, присвоенные константам при их инициализации, не могут быть изменены в ходе выполнения программы.

В C++ существует и другой способ описания констант, доставшийся в наследство от языка C: с помощью макроопределений использующих директиву препроцессора #define. Например,

```
#define TOP 12
//объявлена константа с именем TOP, равная 12
```

(Точка с запятой после директивы препроцессора не ставится.)

Каждая директива #define позволяет определить одну константу, имя которой следует за директивой. Принято писать имя константы заглавными буквами, что не является требованием компилятора. В отличие от предыдущего способа тип такой константы неизвестен. Процессор при трансляции программы просто заменяет имя константы на определенной с помощью директивы значение.

Данная директива в языке C использовалась также для определения макросов (макроопределение с аргументом). Макросы являются просто текстовыми подстановками и могут не давать компилятору достаточной информации в желательном представлении данной величины. Часто встречаемой ошибкой была, например, такая

```
#define SUMMA(a,b)   a+ b
double result, x=5.2, y=0.8,
```

```
result=SUMMA(x,y)*10;
```

После работы препроцессора получим подстановку вида:

```
result=x+y*10;
```

Для корректной подстановки рассмотренная директива должна выглядеть следующим образом:

```
#define SUMMA(a,b) ((a)+(b))
```

В C++ для определения встраиваемой функции используется ключевое слово inline. Определение аналогичной функции в программе на C++ будет выглядеть так:

```
inline double SUMMA(double a, double b)
{ return (a+b); }
```

При определении и использовании встраиваемой функции надо придерживаться следующих правил:

- определение и объявление функции должны располагаться перед первым ее вызовом;
- имеет смысл определять таким способом только маленькие функции;
- компилятор сам решает, является ли данная функция встраиваемой, при этом он руководствуется размером (до 1200 строк), если функция рекурсивна, то встраиваемой является только первый вызов, некоторые компиляторы не допускают использования в встраиваемых функциях операторы цикла и некоторые другие.

Комментарии

Комментариями называются некомпилируемые фрагменты программы. Комментарий используется для пояснения алгоритма или текста программы. Комментарий либо начинается с двух символов «косая черта» («слэш») // и заканчивается переходом на новую строку, либо заключается между символами-скобками /* и */. Внутри комментария можно использовать любые допустимые на компьютере символы, а не только символы алфавита языка C++, так как компилятор комментарии игнорирует.

Библиотечные функции

Некоторые вычисления, такие как извлечение квадратного корня или нахождение модуля, часто используются в программах. Для удобства программиста любой программный комплекс C++ содержит *стандартную библиотеку* (или *библиотеку стандартных функций*) – собрание ранее написанных функций, выполняющих стандартные вычисления.

Библиотека математических функций содержит:

Имя функции	*Параметр*	*Возвращаемое значение*
acos(x)	$-1.0 \le x \le 1.0$	Арккосинус x, в диапазоне от 0.0 до π
asin(x)	$-1.0 \le x \le 1.0$	Арккосинус x, в диапазоне от $-\pi/2$ до $\pi/2$
atan(x)	x	Арктангенс x, в диапазоне от $-\pi/2$ до $\pi/2$
ceil(x)	x	Верхнее значение x (наименьшее целое число $\ge$ x)

cos(x)	x, выраженный в радианах	Тригонометрический косинус x
exp(x)	x	Значение e (2.718…), возведенное в степень x
fabs(x)	x	Модуль x
floor (x)	x	Нижнее значение x (наибольшее целое число ≤ x)
log(x)	x > 0.0	Натуральный логарифм от x
log10(x)	x > 0.0	Десятичный логарифм от x
pow(x, y)	если x=0, y>0; иначе y - целое	Значение x, возведенное в степень y
sin(x)	x, выраженный в радианах	Тригонометрический синус x
sqrt(x)	x>0.0	Корень квадратный из x
tan(x)	x, выраженный в радианах	

Параметрами и результатами перечисленных математических функций являются переменные типа float.

Использовать библиотечную функцию несложно. Требуется разместить в начале программы директиву #include с указанием требуемого файла заголовков math.h. Затем можно обращаться к функции в любом месте программы.

Пример:

```
#include <iostream.h>
#include <math.h>
void main ( )
{       float alpfa, beta;
        alpfa = sqrt(169.41 + fabs( pow( beta, 5));
        cout<< alpfa; }
```

Тема 8. Методы структурного программирования

Блоки или составные выражения

Любое выражение, оканчивающееся точкой с запятой, является оператором. Оператор также может ничего не содержать – так называемый пустой оператор. Пустой оператор обозначается просто точкой с запятой (так как после каждого оператора, том числе и пустого, по требованию синтаксиса языка C++ обязательно ставится точка с запятой). Любой оператор может представлять собой объявление, выполняемый оператор или блок.

Тело функции является примером блока или составного оператора. Блок является последовательностью из нуля или более операторов. Эта последовательность заключена в фигурные скобки. После окончания блока точка с запятой не ставится.

Блок можно использовать везде, где разрешено использование отдельного оператора.

Блоки часто используются в программах, особенно как составная часть

других выражений. Пропуск пары фигурных скобок может кардинально поменять смысл алгоритма программы, повлиять на ее выполнение и полученный результат.

В программе операторы блока обычно располагаются с некоторым отступом, что не является требованием компилятора. Но это позволяет выделять блок и контролировать расположение скобок. Это повышает читаемость программы, и следовательно, контроль ошибок.

Внутри блока разрешается чередовать объявления и операторы, т.к. объявления в программе С++ могут располагаться в любом месте. Необходимо только помнить, что все переменные должны быть объявлены перед использованием.

Базовые конструкции структурного программирования

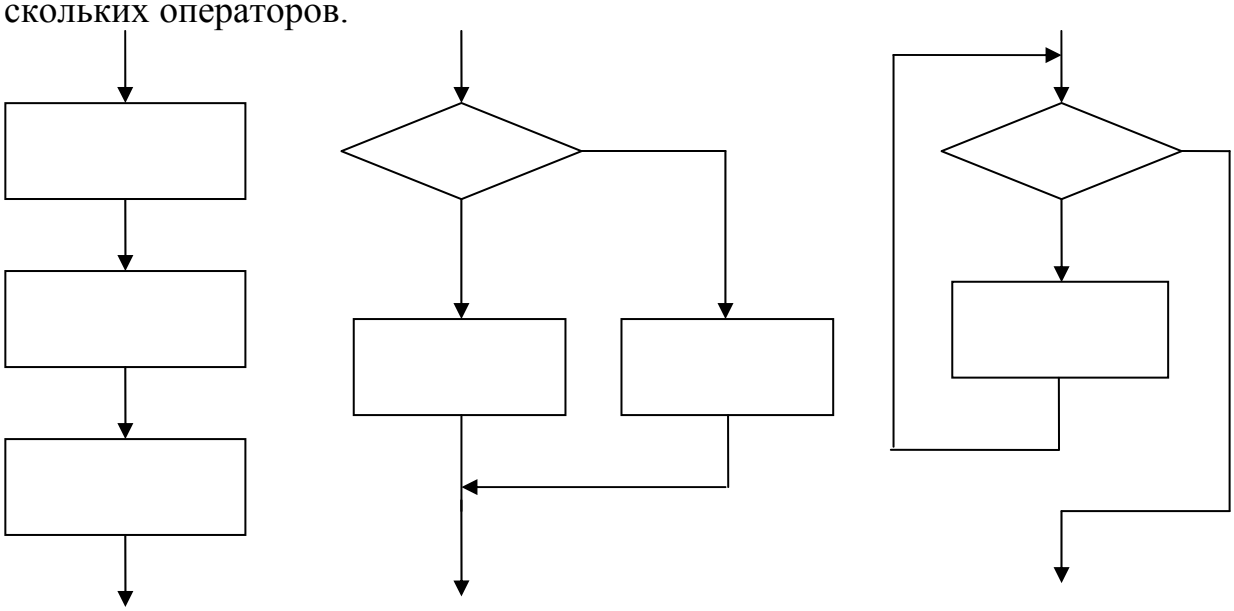
Программу для решения задачи любой сложности можно составить только из трех структур, называемых следованием, ветвлением и циклом.

Следование, ветвление и цикл называют *базовыми конструкциями* структурного программирования.

Следование представляет собой последовательное выполнение нескольких операторов (простых или составных).

Ветвление задает выполнение одного из операторов в зависимости от результата выполнения какого-либо выражения, задающего условие.

Цикл задает многократное выполнение одного или последовательности нескольких операторов.



Следование Ветвление Цикл

Операторы ветвления

Условный оператор if

Условный оператор if позволяет разветвлять вычислительный процесс на два направления. Структурная схема оператора представлена на рисунке.

Синтаксис оператора if:

if (<выражение>) <оператор1>; [else <оператор 2>;]

Ветвь с ключевым словом else не является обязательной. Поэтому она взята в квадратные скобки. (В дальнейшем изложении для обозначения необязательных элементов будут всегда использоваться квадратные скобки.)

Примеры:

```
if ( fvalue>=0.0 )  fvalue = fvalue;  else fvalue = -fvalue;   // вычисляется модуль числа
if ( x<10 ) x+ =10; else x *=2;
if ( f != 0 )  c = 100 / f;
```

Если в какой-либо ветви требуется выполнить несколько операторов, их необходимо заключить в блок (операторные скобки { }), иначе компилятор не сможет определить окончание ветвления.

```
if ( a ) { a++;  v=60*a; }  else v = a;
if ( e==1000 ) { e /=10;  cout << "e= " << e; }  else { e =10+y*y;  y++; }
```

При использовании блока в условном операторе точка с запятой после правой фигурной скобки блока не ставится. Точка с запятой применяется для завершения простых операторов.

Блок может содержать любые операторы, в том числе и другие условные.

```
if ( a<b ) { if ( a<c ) m = a;  else m = c; }  else { if ( b<c ) m = b;  else m = c; }
```

Необходимо помнить, что в этом случае else относится к ближайшему из if. Операторные скобки после первого if необязательны, т.к. в него вложен простой оператор if.

Если требуется проверить несколько условий, их объединяют знакам логических операций.

```
 if ( a<b && ( a>d || a==0 ) )  b++; else { b*=a; a=0; }
```

Записанное условие будет истинно в том случае, если выполнится одновременно условие a<b и одно из условий в скобках. Если опустить внутренние скобки, будет выполнено сначала логическое И, а потом ИЛИ.

Распространенной ошибкой является неверная запись условия проверки переменной на принадлежность диапазону. Например, чтобы проверить условие 0<x<10 нельзя записать его в условном операторе непосредственно, следует писать if (0< x && x>10).

Оператор выбора switch

Оператор switch (переключатель) предназначен для разветвления процесса вычислений на несколько направлений.

Синтаксис оператора switch:

```
switch ( выражение) {
case <константное выражение 1> : <операторы1>; [ break; ]
case <константное выражение 2> : <операторы2>; [ break; ]
…
case <константное выражение n> : <операторы n>; [ break; ]
[default : операторы];
}
```

Выполнение оператора начинается с вычисления выражения, которое должно быть целочисленным. Затем управление передается первому оператору из спи-

ска, помеченному константным выражением, значение которого совпало с вычисленным.

После этого, если выход из переключателя явно не указан (отсутствует break), последовательно выполняются все нижележащие ветви.

Выход из переключателя обычно выполняется с помощью оператора break или return. Оператор break выполняет выход из самого внутреннего из объемлющих его операторов. А оператор return выполняет выход из функции, в которой он описан.

Все константные выражения, расположенные после case, должны быть различны. Если совпадения ни с одним оператором не произошло, выполняются операторы, расположенные после ключевого слова default.

Ветвь default может отсутствовать. В этом случае выполнение программы передается следующему за switch оператору.

Пример предыдущий раздела (печать названия месяца по порядковому номеру) может быть переписан с использованием switch следующим образом:

```
#include <iostream.h>
int main ( )
{ int x;
  cout << “Введите номер месяца” << endl;
  cin >> x;
  switch (x ) {
        case 1 : cout << “Январь”<< endl;          break;
        case 2 : cout << “Февраль”<< endl;         break;
        case 3 : cout << “Март”<< endl;            break;
…
        case 12 : cout << “Декабрь”<< endl;        break;
        default : cout << “Неверный номер” << endl;     }
return 0;
}
```

Несколько меток могут следовать подряд, предполагая выполнение одинаковых действий. Программа вывода времени года по номеру месяца:

Инструкции перехода

В языке C++ имеется четыре инструкции перехода: goto, break, continue, return.

Оператор безусловного перехода goto имеет формат:

goto <метка>;

Тогда в теле той же функции должна присутствовать ровно одна конструкция вида:

<метка> : оператор;

Метка является обычным идентификатором. Оператор goto передает управление на помеченный оператор.

Наличие конструкции безусловного перехода рассматривается как плохой стиль программирования, и считается, что в четко структурированной и грамотно написанной программе этой конструкции быть не должно.

Использование данного оператора оправдано в случаях:

- принудительного выхода вниз по тексту программы из нескольких вложенных циклов или переключателей;
- перехода из нескольких операторов функции на один.

Инструкция break используется внутри циклов или переключателей и позволяет переход в точку программы, находящуюся непосредственно за оператором, внутри которого находится. Таким образом, break обеспечивает вывод из цикла еще до того, как условие цикла станет ложным. По своему действию она напоминает команду goto, только в данном случае не указывается точный адрес перехода. Управление передается первой строке, следующей за телом оператора.

Инструкция continue заставляет программу пропустить все оставшиеся строки цикла, но сам цикл при этом не завершается. Для решения некоторых задач удобно комбинировать инструкции break и continue.

Иногда необходимо прервать выполнение программы задолго до того, как будут выполнены все ее строки. Для этого используют return, которая завершает выполнение той функции, в которой она была вызвана. Если же вызов произошел в функции main( ), то завершается сама программа.

Операторы цикла

Операторы цикла используются для организации многократно повторяющихся вычислений. Любой цикл состоит из тела цикла, т. е. операторов, которые повторяются несколько раз, начальных установок, модификации параметра цикла и проверки условия продолжения выполнения цикла.

Один повтор выполнения операторов тела цикла называется *итерацией*. Проверка условия выполнения цикла производится на каждой итерации.

Параметрами цикла называются переменные, изменяющиеся в теле цикла и используемые при проверке условия продолжения цикла, называются параметрами цикла.

Начальные установки могут явно не присутствовать в программе, их смысл состоит в том, чтобы до входа в цикл задать значения переменным, которые в нем используются.

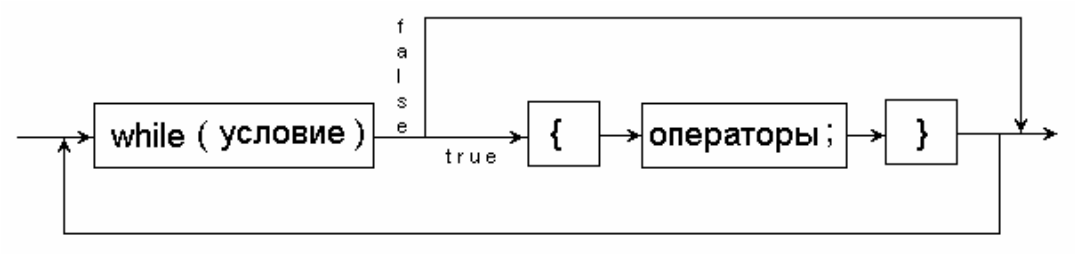
Цикл завершается, если условие его выполнения не выполняется. Возможно принудительное завершение, как текущей итерации, так и тела цикла. Для этого используются конструкции перехода.

Для удобства в C++ существуют три разных оператора цикла – while, do while, for.

Цикл с предусловием (while)

В цикле с предусловием проверка условия продолжения цикла выполняется перед телом цикла.

Цикл с предусловием реализован в C++ оператором цикла while, структурная схема которого представлена на рисунке.



Выражение, стоящее в круглых скобках, определяет условие повторения тела цикла, представленного простым или составным оператором. Если оператор простой операторные скобки { } не ставятся.

Выполнение оператора цикла начинается с вычисления выражения, стоящего в условии. Если оно истинно, выполняется тело цикла. Если при первой проверке выражение ложно (false), цикл не выполнится ни разу.

Тип выражения должен быть арифметическим или приводимым к нему. Выражение вычисляется перед каждой итерацией цикла.

Пример. Программа печатает таблицу значений функции $y = x^3 - x$ с определенным шагом во вводимом диапазоне.

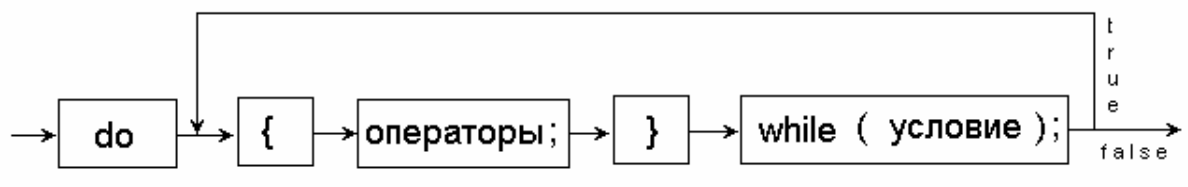
```
#include <iostream.h>
#include <iomanip.h>
void main ( )
{ float x1, xn, d;
  cout << " введите диапазон и шаг изменения x"<< endl;
  cin >> x1 >> xn >> d;
  cout << "|       x       |       y      |" << endl;    // шапка таблицы
  float x = x1;
  while ( x <=  xn )
        { cout << "|" << setw(6) << setprecision(3) << x << "|";
          cout << "|" << setw(6) << setprecision(3)<< x*x*x – x << "|" << endl;
          x+=d;
        }
}
```

Цикл while обычно используется в тех случаях, когда число повторений заранее неизвестно.

Цикл с постусловием (do while)

Тело цикла с постусловием всегда выполняется хотя бы один раз, после чего проверяется, надо ли его выполнять еще раз.

Цикл с постусловием реализован в С++ оператором цикла do … while и имеет вид:



Сначала выполняется простой или составной оператор, составляющий тело цикла, а затем вычисляется выражение, составляющее условие выполнение цикла. Даже если условие заведомо ложно, цикл выполнится один раз. Если условие истинно, тело цикло выполнится еще раз. Цикл завершается, когда выражение станет равным false, или в теле цикла будет выполнен оператор передачи управления.

Пример. Программа угадывания загаданного числа.

```
#include <iostream.h>
#include <stdlib.h>
void main ( )
{  float x, y;
   randomize( );                  // инициализация счетчика случайных чисел
   y = random (10);               // выбор числа в диапазоне от 0 до 10
   do
   {   cout << " введите произвольное число меньшее 10" << endl;
       cin >> x;
       if ( x = = y) { cout <<" вы угадали!"; break; }
       else        if  ( x< y ) cout<< "введите меньшее"<<endl;
                   else    cout<< "введите большее"<<endl;
     }  while (x!=y);
}
```

В приведенном примере для задания числа используется генератор случайных чисел. Оператор randomize( ); инициализирует счетчик случайных чисел. Оператор y = random (10); присваивает переменной y случайно выбранное число из диапазона от 0 до 10. Для использования счетчика случайных чисел требуется подключить библиотеку stdlib.h.

Необходимо отметить, что счетчик случайных чисел генерирует только целые положительные значения из указанного диапазона.

Если требуется, чтобы переменная принимала случайным образом как положительные, так и отрицательные значения, можно воспользоваться конструкцией вида y = random (10) - 5;

Цикл с параметром (for)

Цикл for называют также циклом с заданным числом повторений. Он имеет следующий формат:

for (инициализация; выражение (условие) ; модификации) оператор;

Инициализация используется для объявления и присвоения начальных значений величинам, используемым в цикле. Инициализация выполняется один раз перед выполнением тела цикла.

Цикл с параметром реализуется как цикл предусловием. Выражение определяет условие выполнения цикла: если его результат равен истине, то цикл выполняется. Модификации выполняются после каждой итерации цикла и служат обычно для изменения параметра цикла.

Тело цикла представляет собой простой или составной оператор.

Любое из трех выражений, указанных в скобках, является необязательным. Точки с запятой должны всегда оставаться на своих местах, даже в случае, если все три выражения отсутствуют.

Примеры

```
for ( ; ; );      // пример бесконечного цикла
for (y=2; y<20; y++)  r+=y;
```

В инициализации и модификации параметра можно писать несколько операторов, разделенных запятой.

```
for (i = 1, s=1; i<11; i++) s*=i;          // вычисления факториала 10
```

Цикл for обычно используется в тех случаях, когда можно точно определить необходимое число повторов.

Допускается объявление переменной прямо в строке инициализации цикла for. Значение счетчика цикла может не только увеличиваться, но и уменьшаться, причем на произвольный шаг, который может быть не только целым, но и числом с плавающей точкой.

```
for (float k = 11.5; k>0.5; k - = 0. 5) s+=k;
```

В качестве параметра цикла можно использовать и символьную переменную.

```
for (char ch='a'; ch< 'z'; ch++) …
```

Если тело цикла содержит более одной команды, следует использовать фигурные скобки и руководствоваться определенными правилами оформления, чтобы сделать текст более наглядным.

```
#include<iostream.h>
#include<math.h>
#include<iomanip.h>
void main( )                                  // табулирование функции
{ cout << "+----+-----------+ \ n" ;
  cout << "|  T  |      Y      | \n";
  cout<< "+-----+-----------+ \n";
  int n=10, t;
  float a=0.3, y;
  for ( t=1; t <=10;+ +t )                                  // тело цикла
        {   if ( sin( ( t * t + 1 ) / n )>0 )   y=a*sin( ( t * t+1) / n );
            else          if (sin( ( t * t + 1 ) / n ) < 0 )  y=cos( t+ 1/n );
            cout<< "|  " << setw(2)<< t << " |  "<< setw(5) << setprecision(2) << y << "    | \n";
        }
 cout << "+ -----+ -----------+ \n";
}
```

Массивы

При использовании простых переменных каждой области памяти для хранения данных соответствует свое имя. Если с группой переменных одного типа требуется выполнить различные действия, им дают одно имя и отличают по порядковому номеру. Это позволяет записывать множество действий и операций над этими элементами через циклы.

Конечная именованная последовательность однотипных величин называется массивом. Для создания массива компилятору необходимо знать тип данных, количество элементов в массиве и требуемый класс памяти. Массивы могут иметь те же типы данных, что и простые переменные.

Синтаксис объявления массива:

<Тип данных> <имя массива> [размерность массива];

Примеры:

```
int array[45];                  //массив из 45 элементов целого типа с именем array
double rt[12];                  // массив rt, состоящий из 12 элементов типа double
const int ARRAY_SIZE=90;
float alp[ARRAY_SIZE];      // вещественный массив alp из 90 элементов
```

Размерность массива предпочтительнее задавать с помощью именованных констант, как это сделано в примере. При таком подходе для изменения во всей программе достаточно изменить значение константы в ее объявлении.

Возможна также инициализация массива при его объявлении. Для этого значения элементов массива перечисляются через запятую в фигурных скобках после имени массива.

```
int a[5]={4, 90, 71, 45, 3};
```

Количество элементов должно соответствовать размеру массива. Если их окажется меньше, то недостающие элементы будут инициализированы нулями (или мусором, хранящимся в памяти). Если же элементов окажется больше – компилятор выдаст ошибку.

При инициализации допускается использование пустых скобок в объявлении массива, и тогда компилятор сам определяет размерность массива. Например,

```
float x[ ]= {4, 8, 19};  // размерность x равна 3
```

Для доступа к элементу массива после его имени указывается номер элемента (индекс), заключенный в квадратные скобки. Нумерация элементов массива начинается с нуля. Следовательно, диапазон значений для объявленного в примере массива x лежит в пределах от 0 до 2. Следовательно,

x[0]=4 x[1]=8 x[2]=19

В памяти компьютера элементы массива располагаются последовательно друг за другом. Место в памяти для хранения элементов массива выделяется компилятором сразу после обработки его объявления.

Так объявление вида

```
float y[5]= {0.7, 1.9, 8.4, 3.1 };
```

выделит место для расположения пяти элементов типа float. Так как выполнена инициализация четырех первых элементов, они сразу займут свое место в памяти компьютера, а ячейки памяти для пятого элемента пока останутся свободными:

0. 7	1. 9	8. 4	3. 1	
y[0]	y[1]	y[2]	y[3]	y[4]

В C++ не осуществляется проверка значений индексов на выход за пределы массива, ни в процессе компиляции, ни в процессе выполнения программы. Так

если, в тексте программы напишем выражение

```
y[5]=2;
```

компьютер сохранит значение 2 в ячейке памяти, следующей за ячейкой, соответствующей последнему элементу массива. При этом старое значение данной ячейки будет затерто. Поэтому проверка значения индекса – обязанность программиста.

К элементам массива можно применять все операции, допустимые для обычной переменной типа, соответствующего типу элементов массива. Можно присваивать ему значения

```
y[0]=56;
```

применять арифметические операции

```
y[2]=7*y[0]+3;
```

считывать или выводить значения

```
cin >> y[0];    cout << y[1];
```

передавать его в качестве параметров функций

```
x = sqrt (y[1]);
```

Пример: Программа подсчета суммы элементов массива.

```
#include<iostream.h>
void main ( ) {
  const int n=10;
  int i, m[n], s=0;
  cout << " введите" << n << "чисел" <<endl;
                  // ввод элементов массива с клавиатуры
  for ( i=0; i < n; i++) cin >> m[ i];
  for ( i=0; i < n; i++) s+=m[ i ] ;
  cout << "сумма введенных элементов равна "<<s<<endl;
}
```

Для улучшения эффективности алгоритма ввод элементов массива и подсчет их суммы можно объединить в один цикл:

```
  for ( i=0; i<10; i++) { s+=m[i];    cin >> m[i]; }
```

Пример: Программа выводит на экран количество дней в месяце для невисокосного года.

```
include< iostream. h>
int days[ ]={31,28,31,30,31, 30, 31, 31, 30, 31, 30, 31};
void main ( )
{
for (int index=0; index<sizeof(days)/(sizeof(int)); index++)
        cout << "месяц" << index+1 << " имеет" << days[index] << "дней \n";
}
```

Размерность массива определяется компилятором, а в цикле определение размерности массива происходит с использованием операции sizeof.

Работа с массивами

Классическими задачами при работе с одномерными массивами являются нахождение суммы и произведения элементов массива, определение минималь-

ного и максимального элементов массива, сортировка (упорядочивание) элементов.

Пример нахождения суммы элементов массива рассмотрен в предыдущем разделе. Суммирование элементов происходит в цикле, но необходимо помнить, что значение переменной, предназначенной для хранения значения суммы, должно обязательно равно нулю до начала цикла. При подсчете произведения элементов массива значение произведения до цикла должно быть равно единице.

```
#include<iostream.h>
void main ( ) {
  const int n=15;
  int i, m[n], p=1;
  cout << " введите" << n << "чисел" <<endl;
                  // ввод элементов массива с клавиатуры
  for ( i=0; i < n; i++)
         { cin >> m[ i];
           if ( m[ i] > 0 ) p*= m[ i ] ; }
  cout << "произведение положительных элементов равно "<<p<<endl;
}
```

Стандартный алгоритм нахождения минимального значения заключается в следующем: предполагаем, что первый элемент является минимальным. Далее в цикле каждый следующий элемент сравниваем с минимальным. Если он меньше минимального (обнаруженного на данный момент), то минимальным становится текущий элемент.

Для реализации данного алгоритма нужно не забыть ввести вспомогательную переменную для хранения значения минимального элемента.

Пример. Нахождение минимального элемента в массиве, состоящем не более чем из 50 элементов.

```
#include<iostream.h>
void main ( )
{const int n=50;
  int i, k,  m[n], min;
 cout << " сколько элементов будете вводить?" <<endl;
                  // задаем количество элементов массива с клавиатуры
 cin >> k;
                  // ввод элементов массива с клавиатуры
  for ( i=0; i< k; i++)
         { cout << " введите m[ " << i << " ] ";  cin >> m[ i ];}
  min = m [ 0];
  for ( i=1; i< k; i++)  if ( m[ i ] < min )   min=m [ i ];
  cout << "минимальный из введенных элементов равен "<< m[ i] << endl;
}
```

Наиболее часто встречающая ошибка при создании такой программы заключается в том, что студенты забывают, что записи вида min = m [i]; и m[i] = min; не эквивалентны. В первой записи переменной min присваивается значение m[i] при этом само m[i] остается без изменения. Во втором случае наоборот.

Для нахождения максимального элемента нужно в условном операторе знак меньше поменять на знак больше. Рассмотрим пример нахождения максимального значения и его порядкового номера.

```
#include<iostream.h>
void main ( )
{
const int k=10;
  int i,   m[ k ], max, nmax;
  for ( i=0; i< k; i++)
        { cout << " введите m[ " << i << " ] ";  cin >> m[ i ]; }
  max = m [ 0]; nmax=0;
  for ( i=1; i< k; i++)  if ( m[ i ] > max )  { max=m [ i ]; nmax=i };
  cout << "максимальный элемент равен "<< m[ i] << "его номер" << nmax;
}
```

Многомерные массивы

Для объявления многомерного массива необходимо после его имени задать несколько размеров, заключенных в квадратные скобки. Размерность массивов в С++ не ограничивается. Она может зависеть от возможностей компьютера и особенностей конкретного компилятора. Как правило, на практике используются двумерные массивы.

Двумерный массив в С++ представляет собой массив одномерных массивов.

```
float dam[4] [5];
```

В этом случае массив будет содержать 4 строки и 5 столбцов. Инициализация двумерного массива происходит следующим образом:

```
float art [5] [2]={  { 1. 2, 1. 5 },
                     { -4. 0, 3. 6 },
                     { 2. 3,  -6.1 },
                     { 7. 3,  0. 4 },
                     { 0. 0, -2. 7 }  };
```

При этом внутренние фигурные скобки могут быть опущены. Например,

```
float art [5][2] = {1. 2, 1. 5, -4. 0, 3. 6, 2. 3, -6. 1, 7. 3, 0. 4, 0. 0, -2. 7} ;
```

Все сказанное может быть распространено на трехмерный массив.

```
int rum [3][7][2]; // массив размерности три, каждый элемент которого двумерный массив.
```

Если одномерный массив используется для представления списка, то двумерный массив – для представления таблицы, содержащей строки и столбцы. При этом подразумевается, что все элементы принадлежат одному типу данных. При обращении к отдельному элементу двумерного массива нужно указать его позицию в строке и столбце.

Обработка данных в двумерных массивах означает обращение к массиву одним из четырех способов: случайным образом, по строкам, по столбцам, по всему массиву. Каждый из этих способов может включать обработку части массива.

Самым простым способом получения значения элемента массива является точное указание местоположения элемента. Такой процесс называется случайным

доступом, потому что пользователь может ввести любую случайную комбинацию координат, например

```
cout<< art [0][4];
```

Работа со строками

Часто встречаются ситуации, когда требуется обратиться к элементам массива в определенном порядке (например, найти максимальный элемент в каждой строке матрицы).

Пусть задан двумерный массив, содержащий 5 строк и 6 столбцов:

```
int A[5] [6];
```

Предположим, что нужно сложить все элементы строки с номером 3 (четвертая строка) в массиве А и вывести результат. Это можно легко сделать с помощью цикла for:

```
total = 0;
for ( int col = 0; col < 6; col++ )       total+= A[3][col];
        cout << "результат равен" << total << endl;
```

Этот цикл проходит по всем столбцам массива А, но номер строки всегда остается равным 3. Каждое значение из этой строки прибавляется к переменной total.

Если нужно получить сумму в двух строках – номер 1 и 2, то конечно можно дважды привести предыдущий фрагмент дважды: в первый раз указывать индекс 1, а во второй – 2.

```
total = 0;
for ( int col = 0; col < 6; col++ )       total += A[1] [col];
cout << "результат равен" << total << endl;
total = 0;
for ( int col = 0; col < 6; col++ )       total += A[2] [col];
cout << "результат равен" << total << endl;
```

Но правильнее создать вложенные циклы, а индекс строки сделать переменной - параметром внешнего цикла.

```
for ( int row = 1; row < 3; row++)
        { total = 0;
          for ( int col = 0; col < 6; col++ )     total +=A [row] [col];
           cout << "результат" << row << " строки равен" << total << endl;
        }
```

Этот способ короче и его значительным преимуществом является несложная его модификация для случая обработки любого диапазона строк.

Внешний цикл управляет изменением номера строки, а внутренний – номером столбца. Для каждого значения переменной row обрабатываются все столбцы, затем внешний цикл переходит к следующей строке.

Таким образом, обращение к элементам массива производится в следующем порядке:

A[1][0] A[1][1] A[1][2] A[1][3] A[1][4] A[1][5]

На второй итерации внешнего цикла переменная row увеличится на единицу и становится равной 2, а индекс столбца изменяется от 0 до 5:

A[2][0] A[2][1] A[2][2] A[2][3] A[2][4] A[2][5].

Таким способом можно обрабатывать все элементы в таблице.

В программах для обозначения строк очень часто используется переменная i, а для столбцов – j.

Пример. В матрице размером 4x7 определить максимальные элементы каждой строки и записать их в одномерный массив и вывести его на печать.

```
#include <iostream. h>
int main ( )
{ const int I =4;        // количество строк
  const int J =7;        // количество столбцов
   int m[I][J]={ 4, 7, 0, 9, 6, 2, 3, 68, 23, 0, -6, 3, 8, 5, 7, 9, -4, 3, 1, 3, 91, 3, 7, 6, 0, 4, 6, 1};
  int mmax [ I ], max, i, j;
   for (  i = 0; i < I; i++)
        {          max= m [ i ][ 0 ];
                   for (  j = 1; j < J; j++ )
                          if  ( m [ i ][ j ] > max )  max = m [ i ][ j ];
                     mmax[ i] = max;
        }
        for (  i = 0; i < I; i++)    cout << "максимумы  строк равны" <<mmax << "  ";
        }
```

Работа по столбцам

При работе со столбцами внешний цикл организуют по второму индексу массива, а внутренний по первому. Так для нахождения всех положительных элементов в каждом столбца и вывода полученных результатов можно написать:

```
 for ( int  j = 0; j < J; j++)
        { summa = 0;
          for ( int i = 0; i < I; i++ )       if ( m[ i ][ j ] > 0)  m += m [ i ][ j ];
          cout << "результат" << j << " столбца равен" << summa << endl;
        }
```

Сначала складываются все положительные элементы первого столбца, выводится результат, и только потом индекс внешнего цикла меняется и происходит обращение к элементам следующего столбца.

Инициализация таблицы

Как и в случае одномерных массивов, двумерные массивы можно инициализировать при объявлении. Это непрактично, если таблица имеет большую размерность. Для ввода значений с клавиатуры также можно использовать вложенные циклы.

```
        for ( int  j = 0; j < J; j++)
        for ( int i = 0; i < I; i++ )         cin >> m [ i ][ j ];
```

Здесь запись производится построчно, для ее выполнения по столбцам внутренний и внешний циклы нужно поменять местами.

При задании массива произвольным образом более эффективно использование элементов счетчика случайных чисел.

Вывод таблицы

Еще одним случаем обработки таблицы является задача вывода ее содержимого на экран в общепринятом виде (таблицей значений). Как правило, это подразумевает ее построчный вывод:

```
for ( int i = 0; i < I; i++ )
{       for ( int  j = 0; j < J; j++)       cout << m [ i ][ j ]<< “  “;
        cout << endl;
}
```

Внешний цикл позволяет выполнить вывод всех элементов строки, изменением второго индекса во внутреннем цикле, и переход на новую строчку перед каждым изменением параметра внешнего цикла.

Символьные строки

Символьная строка – последовательность одного и более символов, заключенная в двойные кавычки. Для формирования символьных строк, занимающих несколько строк программы, используется комбинация символов \ и \n. Кавычки не являются частью строки, они служат для обозначения ее начала и конца. Строки представляются в виде массива элементов типа char. Число элементов символьного массива на единицу больше числа символов в строке. Это требуется для нуль-символа (/0), который автоматически добавляется в качестве последнего байта в памяти для того, чтобы отмечать конец строки.

Строка может быть строковой константой или строковой переменной (символьным массивом). В обоих случаях можно выводить элементы строки с помощью оператора cout<< до тех пор, пока не встретится нулевой символ.

Например:

```
cout<<”results are:”;
char msg[ ]=”welcome”;
cout<<msg;
```

Инициализировать строку можно всю целиком или посимвольно:

```
char message [10]= {‘c’, ‘o’, ‘o’, ‘б’, ‘щ’, ’e’, ‘н’, ‘и’, ‘e’};
```

Для ввода строк существует несколько возможностей. Первая – использование оператора cin. При чтении вводимых данных этот оператор будет пропускать все предшествующие непечатаемые символы: пробелы и символы перевода строки. Он также автоматически добавляет нуль-символ в конец строки.

Например, если объявлены две строки:

```
char firstStr[31], secondStr[31];          //2 строки из 30 символов и место для ‘/0’
```

а поток ввода выглядит как □□abc□□defgh□□15, тогда с помощью кода

```
cin>> firstStr >> secondStr;
```

символы ‘a’,’b’,’c’, ‘/0’ поместятся в элементы массива с firstStr[0] по firstStr[3], а символы ‘d’, ‘e’, ‘f’, ‘g’, ‘/0’ – в элементы массива с secondStr[0] по secondStr[4].

Таким образом, данный оператор нельзя использовать для ввода строк, содержащих пробелы. Второй его недостаток в том, что если строковая переменная недостаточно велика, чтобы вместить в ней последовательность вводимых симво-

лов и нуль-символ, то данные из потока ввода будут записываться в память за пределами массива.

Для вывода строк также можно использовать функцию printf( ). Для ввода – scanf( ) и get( ). Функция get ( ) в этом имеет два параметра: строковую переменную и целое типа int, отвечающее за количество вводимых символов в строке плюс один (для нуль-символа).

Например:

```
char line[51];
cin.get(line, 51);
```

В этом случае непечатные символы не пропускаются. Функция считывает и сохраняет полностью всю строку ввода (длиной не более чем указано вторым параметром). Чтобы правильно считать две последовательные строки, нужно не забывать о символе передачи строки:

```
char dummy;
cin.get(line1, 51);
cin.get(dymmy);       //убрать '/n' из потока ввода перед использованием get
cin.get(line2, 51);
```

Язык С++ предлагает большой ассортимент полезных функций для работы со строками. Прототипы всех функций работы со строками содержатся в файле string.h. Рассмотрим несколько из них.

Функция strlen() вычисляет длину строки без нуль-символа. Функция имеет один аргумент – имя строки.

```
#include <string.h>
#include <iostream.h>
void main ( )
{
char str[ ]= "Hello, my friend!";
cout<<strlen(str);              // результат 17
}
```

Функция strcmp( ) сравнивает две строки. Она имеет два аргумента и возвращает целое. Значение этой функции равно 0, если строки равны. Значение strcmp(str1, str2) – целое число меньше 0, если str1<str2 и целое >0, если str1>str2. Сравнение строк происходит в лексикографическом порядке, т.е. в порядке их расположения в словаре.

Пример. Функция, проверяющая правильность введенного пароля с трех попыток.

```
#include <string.h>
#include <iostream.h>
int password ( )
{
char s[5], pass[ ]="лето";
int i_true=0;
for (int i=0; i<3; i++)
        { cin>>s;
        if (strcmp (s, pass)= =0)  {  i_true=1; break; }
```

```
        }
if (i_true = = 0) {cout<<"пароль неверен"; return 1; }
else {cout<<"пароль верен"; return 0; }
}
```

Функция strcpy( ) копирует содержимое второй, из указанных в качестве параметра, строки в первую, замещая прежние данные, включая ‘/0’. При этом первая указанная строка должна иметь достаточный размер.

```
char mystr[100];
strcpy(mystr, "abcdefgh");
```

Функции

Функция – это именованная последовательность описаний и операторов, выполняющая какое-либо законченное действие. Функция может принимать параметры и возвращать значение.

Фигурные скобки отмечают начало и конец тела функции. В круглых скобках после ее имени, в общем случае, содержится информация, передаваемая этой функции – описание параметров (тип и имя).

Наличие списка параметров и их описаний не является обязательным. Но круглые скобки всегда должны присутствовать после имени функции. Тип параметра может быть любым. Если параметров несколько, их описания разделяются запятыми. Если функции не передаются величины, то вместо списка аргументов можно задавать ключевое слово void.

Некоторые компиляторы требуют обязательного указания типа возвращаемого результата перед именем функции. Если тип результата не указывается, то предполагается, что функция возвращает значение типа int. Если функция не возвращает результат, указывается слово void.

Возвращает значение оператор return, с помощью которого можно передать в вызывающую функцию только одно значение. Возвращаемое значение берется в круглые скобки после ключевого слова return. Круглые скобки не являются обязательными, но считаются правилом хорошего тона среди программистов.

Переменные, используемые в функции, отличные от параметров, должны описываются внутри тела функции. Они называются локальными.

Пример. Функция, возвращающая минимальное значение из трех величин.

```
        // определение функции min
float min (float val1, float  val2, float val3 ) {
        if ( val1< val2 &&  val1<val3) return val1;
                else if ( val2 < val1 &&  val2<val3) return val2;
                        else if ( val3 < val2 &&  val3<val1) return val3;
}
```

Эта функция может быть вызвана любой другой функцией. Вызов может выглядеть, например, так

```
 a = min (525, 675, 819);
```

Если в программе объявлены некоторые переменные: `float a, a1, a2, a3;` тогда возможен вызов

a = min (a1, a2, a3);

Параметры (аргументы) функции, используемые в ее описании, называются *формальными*, параметры, содержащиеся в вызове функции и располагаемые на их месте, называются *фактическими*.

Так в описании функции min формальными параметрами являются val1, val2, val3, а фактическими параметрами в ее вызове становятся 525, 675, 819 и a1, a2, a3.

Количество и типы формальных и фактических параметров должны совпадать. Иначе компилятор выдаст ошибку.

В случаях, когда вызываемая функция не возвращает значение, ее вызов представляет собой просто имя функции, за которым в круглых скобках указываются фактические параметры, если таковые имеются.

Функции могут располагаться в тексте программы в любом порядке, но следует помнить, что они должны быть определены до своего вызова.

Так пример программы, содержащей функцию min( ) может выглядеть так:

```
#include <iostream.h>
// определение функции min
float min (float val1, float  val2, float val3 )
{                                                          // начало функции min
        if ( val1< val2 &&  val1<val3) return val1;
              else if ( val2 < val1 &&  val2<val3) return val2;
                    else if ( val3 < val2 &&  val3<val1) return val3;
}                                                          // окончание функции min
// определение функции main
void main ( )
{                                                          // начало функции main
  int x, y, z;
  cout<< " Введите три числа";
  cin>> x >> y >> z;
  cout<< "минимум из этих чисел"<< min (x, y, z);   //вызов функции min
}                                                          // окончание функции main
```

Необходимо отметить, что приведенный пример является "неграмотным" с точки зрения эффективности программы. Поскольку функции в программе должны создаваться с целью упрощения алгоритма программы, структуризации сложной программы, ее наглядности и читаемости. В данной программе нецелесообразно выделять отдельную функцию нахождения минимума, так как это только усложняет ее текст. Функции используются, как правило, когда требуется выполнить одинаковые действия с данными, имеющими различными значениями одинакового типа.

Разделение программы на функции позволяет избежать избыточности кода, поскольку определение функции записывается один раз, а вызывать ее можно многократно из разных точек программы.

Тело функции похоже на любой другой фрагмент кода, за исключением того, что оно содержится в отдельном блоке внутри программы. При разработке функции нужно формально описать ее поведение и механизм взаимодействия с ней, т. е. входные и выходные значения.

Например, с помощью программы, решили вывести сообщение «С Днем рождения!!!», ограничив его сверху двумя строками звездочек, а снизу четырьмя строками. Можно определить функцию вывода на печать двух строк и воспользоваться ею трижды. Данная функция не будет возвращать значение, а в качестве входного параметра можно использовать цвет, которым будут выводиться символы.

```
#include <iostream.h>
#include <conio.h>
void print_str( int x)                    //определение функции print_str
        { textcolor(x );
          cout<< " ******************** "<<endl;
          cout<< " ******************** "<<endl;
        }
void main ( )           // определение функции main
        {
        clrscr( );      // очистка экрана
        textbackground(14 );
        print_str( int 4);
        textcolor(3+128);
        cout<<"С Днем рождения!!!"<<endl;
        print_str( int 10);
        print_str( int 12);
        clrscr( );
        }
```

Нередким является случай, когда функция вызывается до того, как будет объявлена. В этом случае используется прототип функции. Прототип функции имеет следующий вид:

< возвращаемый тип > имя функции (параметры);

Таким образом, прототипом является заголовок функции, оканчивающийся точкой с запятой. В списке параметров обычно указывается тип и имя для каждой переменной; элементы списка разделяются запятыми. Указание имени параметров в прототипе не обязательно, но, как правило, применяется.

Прототип информирует компилятор о существовании функции, о типе возвращаемого значения, а также о типе и количестве аргументов, которые ей передаются.

Так для функции из вышеприведенного примера прототип выглядит следующим образом:

```
        void print_str( int x);
```

или

```
        void print_str( int );
```

Прототипы функций, как правило, располагают после директив препроцессора, до описания основной функции. Рассмотрим пример программы нахождения площадей треугольников с использованием прототипов функций.

```
#include <iostream.h>
```

```
#include <math. h>
void print_area (float, float, float);   //объявление прототипа функции print_area
void main (void)                                  //описание функции main
{
  int n; float a, b, c;
  cout<<"Введите количество треугольников"<<endl;
  cin>>n;
  for (int i=1; i<=n; i++)
        { cout<< "Введите стороны треугольников a, b, c >0" << endl;
         cout<<"a = ";  cin>>a;
        cout<<"b = ";  cin>>b;
        cout<<"c = ";  cin>>c;
        print_area (a, b, c);
        }
}
void print_area (float x, float y, float z)      //описание функции print_area
{
    if ( ( x+ y > z )&&( x+ z > y )&&( y+ z > x ) )
    { float p=( x + y+ z) / 2;
      cout <<"Площадь равна "<<sqrt(p*(p-x)*(p-y)*(p-z))<<endl;
    }
    else cout<<"Треугольник невозможно построить"<<endl;
}
```

Нужно разделять понятия «определение» и «объявление» функции. Объявление функции это заголовок или прототип. Определение функции, кроме объявления, содержит тело функции.

Все функции в программе, созданной на языке С++, равноправны, т.е. любая функция (включая main) может вызвать другую, в том числе и саму себя. Вызов функцией самой себя называется рекурсией. В языке С++ количество рекурсивных вызовов не ограничивается, а определяется лишь возможностями компьютера.

Указатели

При обработке объявлений переменных компилятором в памяти выделяется место в памяти в соответствии с ее типом. После чего все обращения в программе к переменной по ее имени заменяются компилятором на начальный адрес области памяти, которая выделена под ее хранение. В ней располагается значение переменной.

Программист может сам определять переменные для хранения адресов памяти. Указатель является переменной, которая содержит адрес другой переменной или функции. Указатель не является самостоятельным типом, он всегда связан с каким-либо другим типом.

Описание указателя определяет тип данных, на которые указатель ссылает-

ся. Синтаксис объявления указателя:

<тип указываемых данных> * <имя указателя>;

Примеры:

```
int *int_ptr;              // указатель на целое
double *d_ptr;             // указатель на тип double
char *c;                   // указатель на символьную переменную
int *array[10];            // объявление массива указателей, каждый из которых
                           // указывает на  значение int
int (*pointer)[10];        // объявлен указатель с именем pointer, который
                           // указывает на  массив из 10 элементов
```

Возможно также использование указателя на тип void, который может указывать на объект любого типа. Его обычно называют пустым указателем. При выполнении операций над пустым указателем, либо над объектом, на который он указывает, необходимо явно привести тип указателя к типу отличному от void.

Как и любые другие переменные, указатели можно инициализировать при их объявлении. Например, в следующем фрагменте создаются две именованные ячейки памяти result и p_result.

```
int result;
int *p_result=&result;
```

Идентификатор result представляет собой обычную целочисленную переменную, а p_result – указатель на переменную типа int. Одновременно с объявлением указателя p_result происходит его инициализация адресом переменной result. Сама переменная result остается неинициализированной.

При определении указателей надо стремится выполнить их инициализацию. Непреднамеренное использование неинициализированных указателей – распространенный вид ошибок в программах.

Можно также записывать инициализатор после имени указателя в круглых скобках:

```
int result;
int *p_result (&result);
```

или явно присваивать указателю адрес области памяти:

```
float* r=  (float*)0xB8000000;
```

где 0xB8000000 – шестнадцатеричная константа.

Для присваивания указателю пустого значения используют 0 или константу NULL, определенную в заголовочных файлах C++.

Операции с указателями

Для указателей, как и для переменных других типов, существует ряд операций: присваивание, сложение с константой, вычитание, инкремент, декремент, сравнение, приведение типа, косвенная адресация или разадресация (разыменовывание) – * и операция получения адреса – &.

Результатом операция косвенной адресации является величина, помещенная в ячейку с указанным адресом. Ее можно использовать для получения и изменения значения некоторой величины.

Операция получения адреса (&) можно применять далеко не с каждым вы-

ражением. Когда за этим знаком следует имя переменной, результатом операции является номер ячейки памяти, в которой она хранится.

Недопустимо попытка получения адреса в случаях:

- константного выражения, например ptr=&57;
- в выражениях с арифметическими операторами

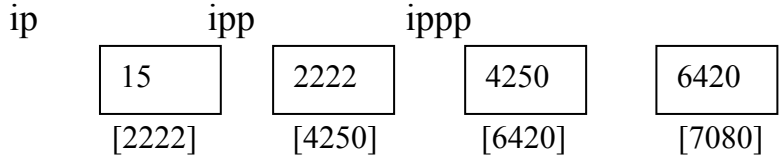
```
int rezult=0;
ptr=&(result+35);
```

- с переменными класса памяти register.

В C++ можно создавать указатели на другие указатели, которые, в свою очередь, содержат имена реальных переменных. Чтобы объявить в программе указатель, который в свою очередь будет хранить адрес другого указателя, нужно просто удвоить число звездочек в объявлении. Количество указателей в цепочке, задающее уровень косвенной адресации, соответствует числу звездочек перед именем указателя. Уровень косвенной адресации определяет, сколько раз следует выполнить операцию раскрытия указателя, чтобы получить значение конечной переменной. В следующем фрагменте создается ряд указателей с различными уровнями косвенной адресации.

```
int value=15;
int *ip;
int **ipp;
int ***ippp;
ip=&value;
ipp=&ip;
ippp=&ipp;
```

В четырех первых строчках объявлены четыре переменные: value типа int, указатель ip на переменную типа int (первый уровень косвенной адресации), указатель ipp (второй уровень адресации) и указатель ippp (третий уровень адресации). Можно создать указатель любого уровня. В пятой строке указателю первого уровня ip присваивается адрес переменной value. Теперь значение переменной value (15) может быть получено с помощью выражения *ip и т.д.



В результате вычитания целого значения указатель будет ссылаться на элемент, смещенный на указанную величину ячеек влево по отношению к текущей ячейке. Соответственно использование операций инкремента и декремента к указателю будет производить сдвиг на количество ячеек памяти соответствующие типу переменной, на которую ссылается указатель. Предполагается, что оба указателя одного типа и связаны с одним и тем же массивом. Иначе результат невозможно предсказать.

Результатом разности указателей будет целое число, соответствующее числу элементов между ячейками, на которые они ссылались.

При записи выражений с указателями нужно помнить о приоритетах операций.

Например,

*ptr++=25;

Операции разадресации и инкремента имеют одинаковый приоритет и выполняются справа налево. Но инкремент используется в постфиксной форме, он выполняется после операции присваивания. Следовательно, сначала по адресу, записанному в ptr, поместится 25, а затем увеличится значение указателя, т.е. аналогично *ptr=25; ptr++;

Указатели и массивы

При использовании указателей происходит работа с адресом ячейки памяти и получение лишь косвенного доступа к ее содержимому.

Часто указатели используются при работе с массивами. Указатели и массивы логически связаны друг с другом. Имя массива является константой, содержащей адрес первого элемента массива. Вследствие чего значение имени массива не может быть изменено оператором присваивания или каким-либо другим оператором. Так, если объявлен массив

float temp[10];

то temp==&temp[0]

Используя операции сложения, вычитания инкремента и декремента к имени массива (или к указателю) можно передвигаться по элементам массива. Т.е. temp++ будет указывать на второй элемент массива, т.е. элемент с индексом 1. Таким образом, можно работать с массивами, не используя их стандартного обращения к собственным элементам. Используя операцию косвенной адресации можно обращаться к значениям элементов массива. Выражение *(temp+1) эквивалентно temp[1].

Это верно и для многомерных массивов. Предположим, есть описание:

int zippo [4][2];

Напомним, что первое число означает количество строк, второе – количество столбцов. Тогда zippo = = &zippo[0][0], a zippo+ 5 на zippo[2][1].

Обращаться к элементу массива можно и следующим способом:

*(zippo[i] + j) или *(* (zippo+ i) + j)

Двумерный массив является одномерным массивом, элементы которого - массивы, следовательно, имя его первой строки zippo[0], а имя четвертой строки - zippo[3]. Однако имя массива является также указателем на этот массив в том смысле, что оно ссылается на его первый элемент. Следовательно,

zippo[0]= =&zippo[0][0]

zippo[2]= =&zippo[2][0].

Передача массива в функцию выполняется следующим механизмом: в описании функции в списке формальных параметров указывается тип элементов массива, его имя, а после пустые квадратные скобки, которые указывают, что данный аргумент является массивом. При вызове функции в списке фактических параметров указывается имя массива. Например,

int sum (int n, int array[ ]) { … } // объявление функции

И тогда, в действительности, функция получает адрес первого элемента массива, и следовательно, она может быть вызвана и следующим образом:

```
int c=sum(15, &arr[0]);        // где int arr[15];
```

или более простой конструкцией:

```
int c=sum(15, arr);
```

В случае передачи в функцию в качестве параметра двухмерного массива в первой паре квадратных скобок размерность не указывается, а во второй паре – скобок должна быть указана обязательно. Например,

```
int sort(int ard[ ][7]);
```

В этом случае фактическим параметром также может служить имя массива.

При передаче в функцию, как одномерного массива, так и двумерного, можно в качестве формальных параметров использовать напрямую указатели, но при этом требуется следить, чтобы не выйти за пределы массива.

Пример.

Программа, вычисляющая значение $z = (a,b) - (b,c)(d,e)$, где a, b, c – векторы размерности 10; a, d и e векторы размерности 12. Запись вида $(x,\ y)$ означает скалярное произведение векторов. Координаты векторов хранятся в одномерных массивах.

```
#include <iostream.h>
#include <stdlib.h>
int scalar (int* x1,  int* x2, int n);
// прототип функции для подсчета скалярного произведения
// два первых параметра – указатели на целый тип,  для передачи массивов
// третий параметр – размерности массивов
void main ( )
{ int a[10], b[10], c[10], d[12], e[12];
int i;
randomize( );
for ( i=0; i< 10; i++)
 {       a[ i ] = random (20) -10;
         b[ i ] = random (20) -10;
         c[ i ] = random (20) -10;
}
for ( i=0; i< 12; i++)
 {       d[ i ] = random (20) -10;
         e[ i ] = random (20) -10;
}
int z = scalar(a, b, 10) – scalar(b, c, 10) * scalar(d, e, 12);
cout << “z=”<<z;
}
int scalar (int* x1,  int* x2, int n)       // описание функции
{ int sc=0;
for ( int k=0; k<n; n++)
sc+= *( x1+ k )* *(x2 + k );
```

```
return sc;
}
```

Поскольку одномерный массив является массивом, состоящим из массивов, о чем уже говорилось, это позволяет использовать функцию, предназначенную для работы с одномерным массивом, для работы со строками двумерного массива.

Пусть описана функция нахождения среднего арифметического массива целых чисел:

```
float mean(int array[ ], int n)
{ int index,  sum=0;
  if (n>0)
                {       for (index=0,sum=0; index<n; index++)    sum+ = *(array + index);
                return (float)sum / n;          // явное приведение типа
                }
  else          { cout<<"Нет массива"<<endl;        return 0;       }
}
```

Параметром функции является целочисленный массив. Поэтому сумма элементов также будет целой. Но в С++ при делении целого на целое результат также целый. Поэтому используется явное приведение типа. Заметим, что к типу float, приводим только первый операнд, и тогда при делении вещественного числа на целое получим вещественный результат. Если привести к типу float все выражение, т.е. (float)(sum / n); - то дробная часть будет равна нулю.

Данную функцию можно использовать для нахождения среднего арифметического строки матрицы:

```
void main ( )
{ int nk[ 3] [4] = { {2,4,6,9}, {10, 20, 40,10}, {3, 7, 0, 9} };
for ( int line=0; line<3; line++)
printf ("Среднее арифметическое %d строки равно %d \n",line+1,mean(nk[line], 4);
}
```

Ссылки

Ссылка это еще одно название для указателя, который не требует разыменовывания при использовании. Разница между указателем и ссылкой заключается в том, что программист может использовать ссылку как обычный объект, несмотря на то, что к объекту будет производиться косвенный доступ, в то время как к указателю необходимо присвоить явно значение адреса объекта.

```
int i = 123;
int *p=&i;
```

Выражение &i является ссылкой на объект с именем i. Через адрес оно ссылается на объект с именем i.

С++ позволяет объявлять ссылочные переменные. Например, объявление &j=i; создает переменную j - ссылку(второе имя) для i .

```
void main(void) {
int i=3;
int
j=2;
```

.....
}

После выполнения присваивания переменная i, также как и j, будет иметь значение 2.

Ссылки безопаснее указателей, поскольку адреса ссылок невозможно переприсваивать. Необходимо помнить что, однажды инициализировав ссылку, ей уже нельзя присвоить другое значение. В отличие от указателей ссылки всегда связаны с объектом. Они используются в качестве переменных, параметров и результатов функций.

Ссылки полезны в функциях, которые возвращают несколько значений. Ведь с помощью оператора return можно возвращать только одно значение. Например, рассмотрим функцию, меняющую между собой значения двух переменных.

```
#include <iostream.h>
void exchange(int &a, int&b)
{ int c=a;
  a=b;
  b=c;
}
void main()
{ int a=100, b=10;
 cout<<"До обмена: a=" <<a<<", b="<<b<<endl;
exchange (a,b);
cout<<" После обмена: a=" <<a<<", b="<<b<<endl;
}
```

Здесь доступ к переменной осуществляется как через ее имя, так и через имя-синоним (ссылку) в вызываемой функции. После завершения вызываемой программы (функции) имя-синоним уничтожается, однако измененное значение переменной в вызываемой функции сохраняется. Таким образом, обращаясь к переменным через ссылки, появляется возможность работать с локальными переменными, как с глобальными. Это позволяет возвращать из функции любое количество значений.

Рекомендации по созданию программы

Главная цель при создании программы с использованием структурного и процедурного методов программирования – получение легко читаемого кода возможно более простой структуры.

Первый шаг состоит в продумывании и записи алгоритма будущей программы на естественном языке или с использованием блок-схемы. Это позволяет продумать алгоритм в деталях, разбить программу на логические блоки, определить их последовательность, продумать комментарии к программе.

Необходимо стараться (если это возможно) разбить алгоритм на последовательность законченных действий. Каждое из них можно оформить в виде функции.

Функция не должна быть слишком большой, как правило, ее текст должен не превышать два экрана.

Если некоторые действия в программе повторяются несколько раз, их также требуется оформить в виде функции, что сделает программу нагляднее и сократит ее размер.

При расположении последовательности функций нужно помнить, что функция может быть вызвана только после ее объявления. Если программа содержит две-три функции, такой порядок достаточно легко определить. Иначе лучше использовать прототипы функций, расположив их в начале программы, сразу после директив препроцессора.

Всю информацию, требуемую для работы функции, необходимо передавать в качестве параметров. В вызове функции строго соблюдать соответствие количества, типов и порядок следования фактических параметров формальным.

Правильный выбор имен улучшают читаемость программы. Для этого существуют несколько рекомендаций. Считается, что несколько первых символов имени должны объяснять содержимое переменной, тем самым, создавая документированность программы. Также популярны, так называемые венгерские конвенции (или венгерская запись имен), которые были разработаны программистом из Microsoft Ч. Симони. В соответствии с ними перед именем указателя следует помещать букву p, перед дальними указателями lp, перед функциями fn.

Обычно, чем больше область видимости переменной, тем ее имя длиннее. Для параметров коротких циклов лучше использовать однобуквенные имена. Имена макросов и констант предпочтительнее записывать заглавными буквами.

Нельзя использовать в качестве имен переменных имена типов и ключевые слова языка.

Желательно инициализировать переменные при их объявлении, а объявлять их как можно ближе к месту непосредственного использования. В небольших функциях удобно все объявления локальных переменных располагать в начале блока.

Если ввод переменных осуществляется через клавиатуру, требуется предварять его выводом сообщения на экран, которое обязательно должно быть информативным. Лучше предусмотреть и правильность ввода переменных. Например, при введении даты нужно проверить, чтобы число не превышало 31, а номер месяца 12.

Использование локальных переменных предпочтительнее глобальных. Если глобальная переменная необходима, лучше объявить ее статической, что ограничит область ее действия одним файлом. Изменение глобальных переменных сложно отслеживать в большой программе.

Не рекомендуется использования в программе чисел в явном виде. Лучше использовать константу. Особенно это актуально в случаях, неоднократного ее использования. Тогда при изменении кода программы, легко изменить ее значение. Константы также должны иметь осмысленные имена.

При использовании ветвления следует избегать проверки лишних условий. Неэффективным кодом считается проверка на равенство нулю. Например, вместо if (k = = 0) лучше писать if (k).

При организации циклов лучше размещать инициализацию и приращения счетчика, проверку условия выхода из цикла в одном месте. Требуется предусматривать аварийный выход из цикла по достижению заданного максимального количества итераций.

Вложенные циклы и блоки должны иметь отступ друг от друга в три-четыре символа, причем блоки одного уровня вложенности должны быть выровнены по вертикали. Желательно, чтобы закрывающаяся фигурная скобка была расположена строго под соответствующей ей открывающейся.

Необходимо проверять коды возврата ошибок и предусматривать печать сообщений в тех точках программы, куда управление программой при ее нормальной работе передаваться не должно. Например, оператор switch должен иметь ветвь default, в случае, когда в нем не перечислены все возможные значения переключателя.

Сообщение об ошибке должно быть информативным и подсказывать пользователю методы ее исправления. Например, при вводе неверного значения в сообщении об ошибке должен быть указан допустимый диапазон.

Написанный программный код нужно тщательно отредактировать. Убрать ненужные фрагменты, сгруппировать описания, оптимизировать проверки условий, проверить условия выхода из циклов, проверить оптимальность разбиения на функции.

Следует сопроводить программу комментариями, которые должны представлять собой правильные предложения без сокращений и со знаками препинания. Но они и не должны подтверждать очевидное. Если комментарий занимает несколько строк, лучше его разместить до комментируемого фрагмента. Абзацный отступ комментария должен соответствовать отступу комментируемого блока. Для улучшения читаемости можно помечать комментарием окончание длинного составного оператора.

Не следует размещать в одной строке программы множество операторов. Важно, чтобы строка программы не выходила за пределы экрана. Между крупными блоками и функция лучше располагать пустую строку.

Для грамотно написанной программы недостаточно подтверждения ее работы и даже получения верного результата. Программа должна иметь четкую структурированность, наглядность, читаемость, сопровождение комментариями, возможность легкой модификации, и желательно, грамотно разработанный, эффективный алгоритм.

Тема 9. Базы данных

База данных (БД) – это поименованная совокупность структурированных данных, относящихся к определенной предметной области (БД по металлургии, БД в деканате о студентах, БД в библиотеке по книгам).

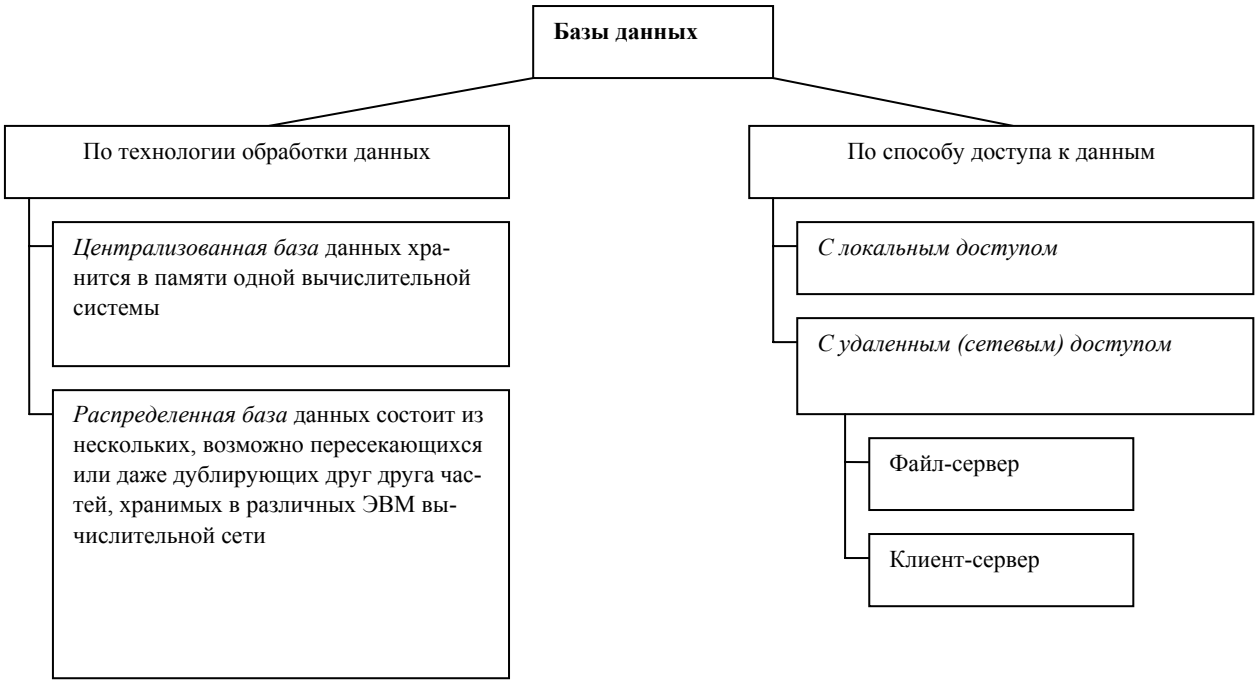
Система управления базами данных (СУБД)– это комплекс программных и языковых средств, необходимых для создания баз данных, поддержания их в актуальном состоянии и организации поиска в них необходимой информации.

Основными функциями системы управления базами данных являются:

- создание структуры базы данных;

- предоставление средств для ее заполнения или импорта данных из другой базы;
- обеспечение возможности доступа к данным;
- предоставление средств поиска, фильтрации, запросов данных.

Классификация БД:



Структурные элементы базы данных:

1. Поле – элементарная единица логической организации данных, которая соответствует неделимой единице информации – реквизиту. Для описания поля используются следующие характеристики:

a) Имя, например, Фамилия, Имя, отчество, Дата рождения;
b) Тип, например, символьный, числовой, календарный;
c) Размер – определяет предельную длину, например, 15 байт;
d) Формат – определяет способ форматирования данных;
e) Маска ввода – определяет форму, в которой вводятся данные в поле;
f) Подпись – определяет заголовок столбца таблицы;
g) Точность для числовых данных, например для десятичного знака для отображения дробной части числа.

2. Запись – совокупность логически связанных полей. Экземпляр записи - отдельная реализация записи, содержащая конкретные значения ее полей.

Запись содержит данные о конкретном экземпляре сущности. База, не имеющая записей, является пустой, но все равно полноценной. Простейший «некомпьютерный вариант» БД – ежедневник, в котором каждому календарному дню выделено по странице. Даже если он пустой, он не перестает быть ежедневником, поскольку имеет структуру.

Файл (таблица) – совокупность экземпляров записей одной структуры.

Типы данных

1. Текстовый – используется для хранения обычного неформатированного текста ограниченного размера (до 255 символов).

2. Поле Memo – для хранения больших текстов (до 64000 символов).
3. Числовой.
4. Дата и время.
5. Денежный – для хранения денежных сумм.
6. Счетчик – для уникальных (не повторяющихся) натуральных чисел с автоматическим наращиванием. Используется для порядковой нумерации записей.
7. Логический – для хранения логических данных, которые могут принимать только два значения, например Да или Нет.

Объекты базы данных

1. Таблицы.

2. Запросы. Служат для извлечения данных из таблицы и предоставления их пользователю в удобном виде. С помощью запроса выполняют отбор данных, их сортировку и фильтрацию, создают новые таблицы.

3. Формы. Являются основным средством создания диалогового интерфейса приложения пользователя. Форма может создаваться для ввода и просмотра взаимосвязанных данных базы на экране в удобном виде.

4. Отчеты. Предназначены для формирования выходных документов вывода данных на принтер. В отчетах приняты меры для группирования выводимых данных и для вывода элементов оформления, характерных для печатных документов.

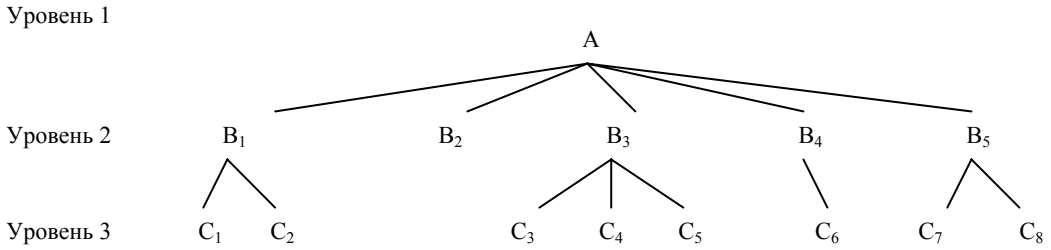
В структуре записи файла указываются поля, значения которых являются ключами: первичными (ПК), которые идентифицируют экземпляр записи, и вторичными (ВК), которые выполняют роль поисковых или группировочных признаков (по значению вторичного ключа можно найти несколько записей).

Виды моделей данных.

Модель данных – совокупность структур данных и операций их обработки.

1. Иерархическая модель данных.

Представляет собой совокупность элементов, связанных между собой по определенным правилам. Объекты, связанные иерархическими отношениями, образуют ориентированный граф (перевернутое дерево).



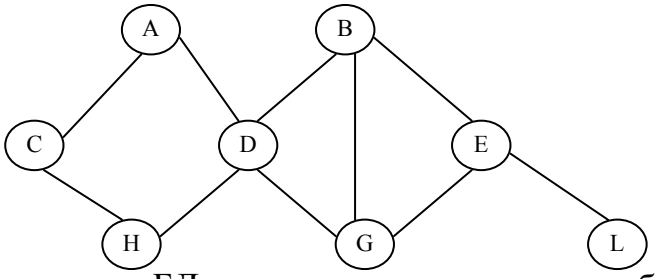
К основным понятиям иерархической структуры относятся: уровень, элемент (узел), связь.

Узел – это совокупность атрибутов данных, описывающих некоторый объект (на схеме – вершины графа). Каждый узел на более низком уровне может быть связан только с одним узлом, находящемся на более высоком уровне.

Примером иерархической структуры БД являются сведения о студентах обучающихся в группе. Можно сказать, что каждый студент учится в определенной (только одной) группе, которая относится к определенному (только одному) институту.

2. Сетевая модель данных.

Объекты в сетевой модели связаны разнородно, т. е. при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим элементом.

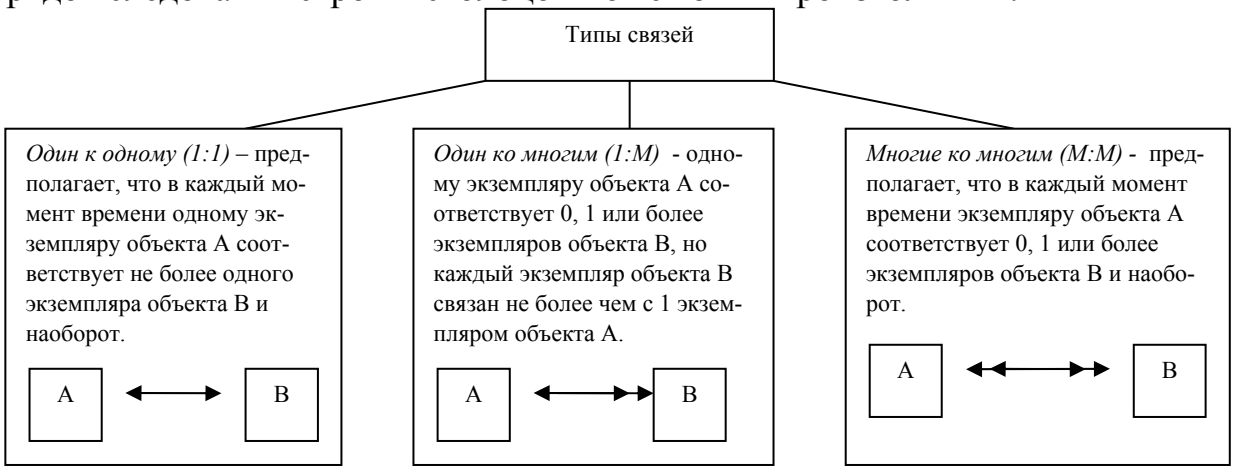


Примером сетевой структуры БД может служить структура базы данных, содержащая сведения о студентах, участвующих в научно-исследовательских работах (НИРС). Возможно участие одного студента в нескольких НИРС, а также участие нескольких студентов в разработке одной НИРС.

3. Реляционная модель данных.

Эта модель ориентированна на организацию данных в виде двумерных таблиц. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы – один элемент данных;
- все столбцы в таблице однородные, т. е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
- каждый столбец имеет уникальное имя;
- одинаковые строки в столбце отсутствуют;
- порядок следования строк и столбцов может быть произвольным.



Гражданин - паспорт Покупатель - торговый агент Студент-преподаватель

Реляционной таблицей можно представить информацию о студентах, обучающихся в вузе.

Поле, каждое значение которого однозначно определяет соответствующую запись, называется простым ключом (ключевым полем). Если записи однозначно определяются значениями нескольких полей, то такая таблица базы данных имеет составной ключ. В примере ключевым полем таблицы является «№ личного дела».

Чтобы связать две реляционные таблицы, необходимо ключ первой таблицы ввести в состав ключа второй таблицы (возможно совпадение ключей); в противном случае нужно ввести в структуру первой таблицы внешний ключ – ключ второй таблицы.

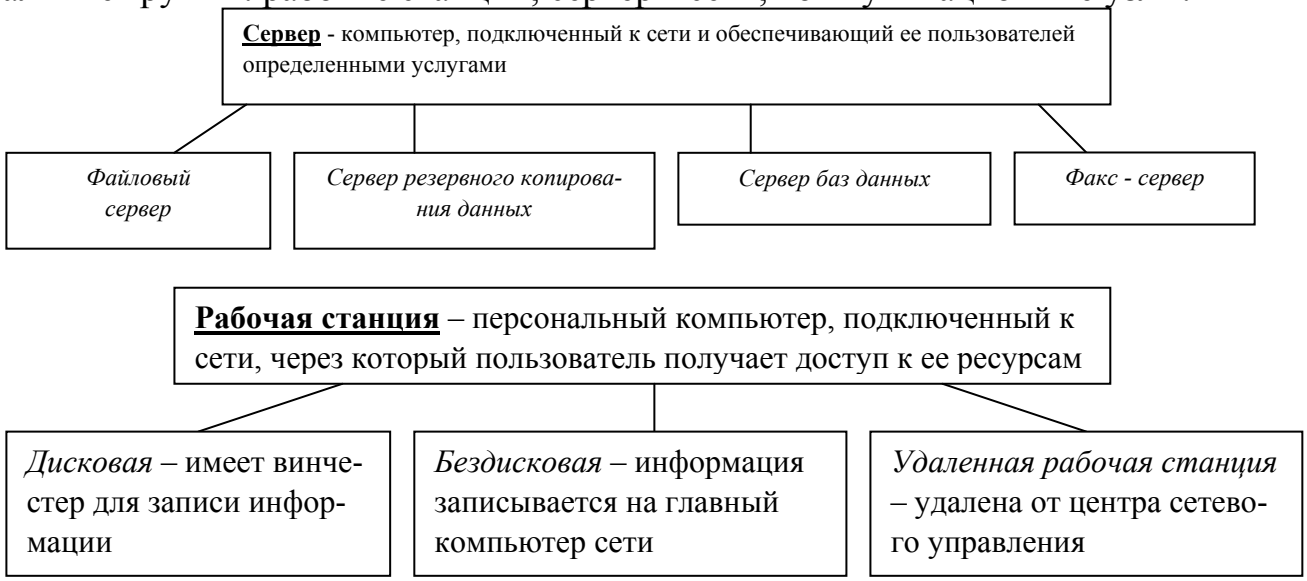
К реляционной модели данных относится Ms-Access.

Тема 10. Компьютерные сети. Интернет.

Компьютерная (вычислительная) сеть - совокупность компьютеров и терминалов, соединенных с помощью каналов связи в единую систему, удовлетворяющую требованиям распределенной обработки данных.

Устройства компьютерной сети

Все устройства, подключаемые к сети, можно разделить на три функциональные группы: рабочие станции; серверы сети; коммуникационные узлы.



К коммуникационным узлам относятся следующие устройства: повторители, коммутаторы (мосты), маршрутизаторы, шлюзы.

Информация передается по сети порциями (пакетами). В сети при передаче отдельного пакета с помощью каналов связи происходит затухание сигнала. Поэтому сеть ограничивают участками (сегментами).

По окончанию каждого участка имеется повторитель – устройство, усиливающее или регенерирующее пришедший на него сигнал. Повторитель, приняв пакет от одного сегмента, передает его дальше. Повторитель осуществляет связывание сегментов.

Коммутатор или мост – это устройство, которое, как и повторитель, выполняет объединение нескольких сегментов.

Маршрутизатор – устройство, соединяющее сети одного или разных типов по одному протоколу данных, анализирует адрес назначения и направляет данные по оптимально выбранному маршруту.

Шлюз – это устройство, позволяющее организовать обмен данными между разными сетевыми объектами, использующими разные протоколы обмена данными.

Основные показатели сети

Производительность сети определяется количеством информации, переданной через сеть или её сегмент в единицу времени.

Надежность работы вычислительной сети определяется надежностью работы всех ее компонентов. Для обеспечения надежности, как правило, информация в сети храниться в нескольких экземплярах (дублирование информации).

Безопасность – это способность сети обеспечить защиту информации от несанкционированного доступа. Задачи обеспечения безопасности решаются применением специального программного обеспечения, использование более безопасной передающей среды в сети.

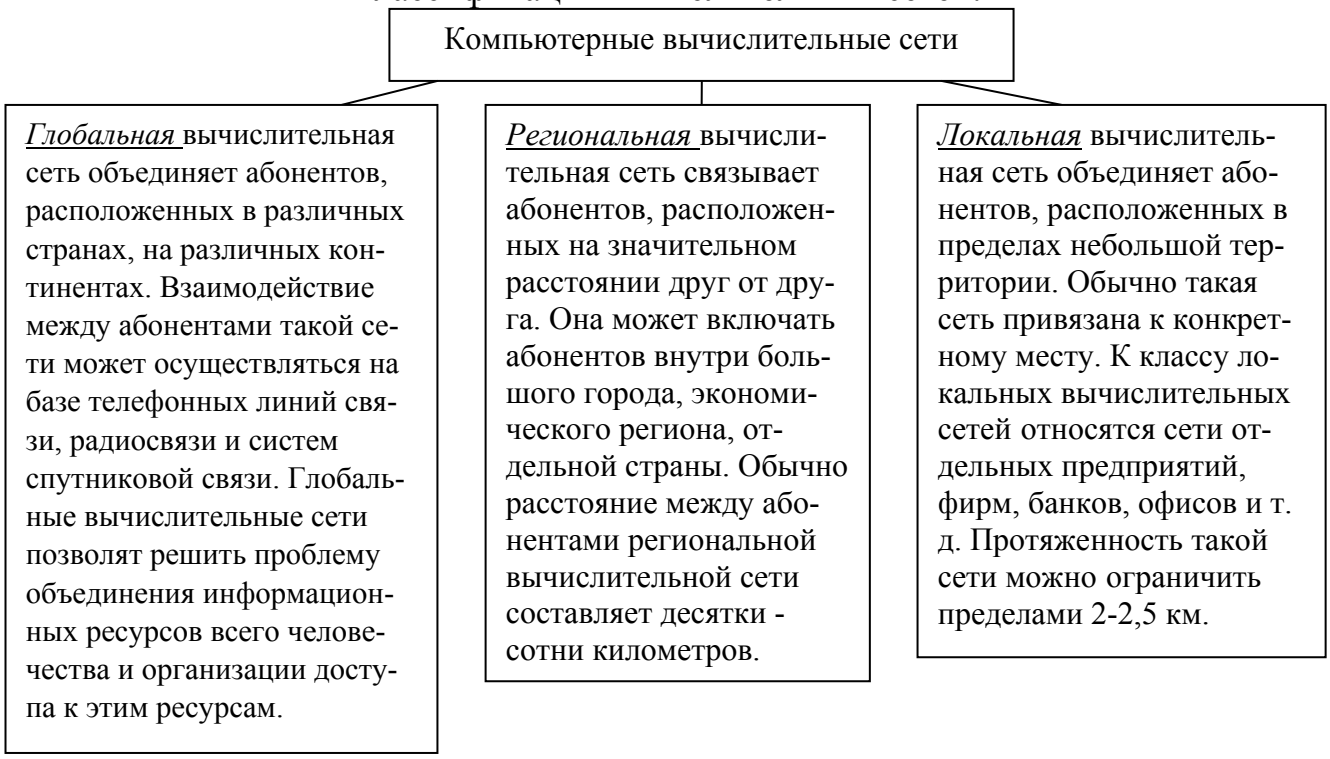
Управлением сетью занимается администратор сети, обычный пользователь не имеет административных прав. Администратор сети должен иметь возможность воздействовать на работу любого элемента сети. Управляемость сетью – возможность определения проблем в работе вычислительной сети или отдельных ее сегментов, выработка управленческих действий для решения выявленных проблем.

Расширяемость определяется возможностью добавления новых элементов сети (рабочих станций, служб).

Масштабируемость сети определяет возможность расширения сети без существенного снижения ее производительности. Например, одноранговые сети обладают хорошей расширяемостью, но плохой масштабируемостью.

Интегрируемость означает возможность подключения к вычислительной сети разнообразного и разнотипного оборудования.

Классификация вычислительных сетей:



Типы передающей среды в компьютерных сетях

В компьютерных сетях используется два основных вида передающей среды: кабельный канал связи и радиосвязь.

Кабельный вид связи

Кабельный вид связи используют преимущественно в локальных, чаще региональных сетях. Кабели бывают нескольких видов: витая пара проводов, коаксиальный кабель, оптоволоконный кабель.

Витая пара состоит из двух изолированных проводов, свитых между собой. Например, телефонный кабель. Такой вид соединения недорогой. Недостаток витой пары - плохая помехозащищенность и низкая скорость передачи информации. Существует неэкранированная витая пара и экранированная витая пара. Дополнительный защитный слой в экранированной паре делает кабель помехоустойчивым, значительно уменьшает электромагнитное излучение.

Коаксиальный кабель по сравнению с витой парой обладает большей прочностью, помехозащищенностью. В центре коаксиального кабеля находится медный проводник, окруженный толстым слоем изоляционного материала. Второй слой сделан в виде оплетки поверх изоляции.

Оптоволоконный кабель не подвержен действию электромагнитных полей. В оптоволоконном кабеле для передачи данных используются световые импульсы. Сердечник такого кабеля изготовлен из стекла или пластика. Сердечник окружен слоем отражателя, который направляет световые импульсы вдоль кабеля путем отражения от стенок. Оптоволоконный кабель более дорогой по сравнению с предыдущими типами.

Радиосвязь

Каналы радиосвязи используют различные диапазоны передачи данных. В радиосвязи передача информации осуществляется от одного ретранслятора к другому. Разновидностью радиосвязи является спутниковая связь, в которой передача данных осуществляется через спутник.

Конфигурация локальных вычислительных сетей

Существуют следующие конфигурации ЛВС: шинная, кольцевая, звездообразная, древовидная. От конфигурации ЛВС зависит, как размещаются абоненты сети и как они соединяются между собой.

Шинной называется такая конфигурация сети, при которой к незамкнутому каналу (шине) с некоторым интервалом подключаются рабочие станции. Информация от абонента-источника распространяется по каналу в обе стороны.

В кольцевой сети информация передается по замкнутому каналу (кольцу), в большинстве случаев только в одном направлении. Каждый абонент непосредственно связан с двумя соседними абонентами, но «прослушивает» передачу любого абонента сети.

Звездообразные сети возникли на основе телефонных сетей с АТС. В центре звездообразной сети находится центральный коммутатор, либо устройство, которое последовательно опрашивает абонентов и предоставляет им право на обмен информацией.

Древовидная конфигурация сети образуется путем подсоединения нескольких простых шин к одной магистральной при помощи мостов или шлюзов.

Системное программное обеспечение вычислительных сетей

Для эффективной работы сетей используются специальные операционные системы. Это сетевые операционные системы, которые устанавливаются на спе-

циально выделенные компьютеры. Признанными лидерами сетевых ОС являются Windows NT, Unix.

Обработка информации в сети распределена между двумя объектами: клиент (рабочая станция) и сервер. Клиент может запросить с сервера чтение – запись файла, поиск информации и т.д. Сервер выполняет запрос, поступивший от клиента. Подобные системы принято называть системами «клиент – сервер».

В зависимости от распределения функций между компьютерами в сети можно выделить одноранговые сетевые ОС и ОС с выделенным сервером.

Одноранговые ОС используются в одноранговых сетях, для которых характерно: нет единого центра управления и единого устройства для хранения данных, сетевая операционная система распределена по всем рабочим станциям. Достоинства: низкая стоимость и высокая надежность. Недостатки: сложность управления сетью, сложность обеспечения защиты информации, трудности обновления программного обеспечения, ограниченность количества рабочих станций.

В сети с выделенным сервером один компьютер выполняет роль сервера. На нем устанавливается сетевая операционная система, к нему подключаются все внешние устройства. Взаимодействие между рабочими станциями в сети осуществляется через сервер. Количество серверов может быть более одного. Достоинства: надежная система защиты информации, высокое быстродействие, отсутствие ограничений на число рабочих станций. Недостатки: высокая стоимость сервера, меньшая гибкость по сравнению с одноранговой сетью.

Любая коммуникационная сеть должна включать следующие основные компоненты: передатчик, сообщение, средства передачи, приемник.

Глобальная сеть INTERNET

Internet представляет собой глобальную компьютерную сеть. Само ее название означает "между сетей". Это сеть, соединяющая отдельные сети.

Internet обеспечивает обмен информацией между всеми компьютерами, подключенными к ней. Тип компьютера и используемая им операционная система значения не имеют.

Существуют компьютеры, самостоятельно подключенные к internet. Они называются хост-компьютерами (host – хозяин).

Каждый подключенный к сети компьютер имеет свой адрес, по которому его можно найти.

Система адресации в Internet

Internet самостоятельно осуществляет передачу данных. К адресам станций предъявляются специальные требования. Адрес должен иметь формат, позволяющий вести его обработку автоматически, и должен нести некоторую информацию о своем владельце.

С этой целью для каждого компьютера устанавливаются два адреса: цифровой IP -адрес(IP - Internetwork Protocol - межсетевой протокол) и доменный адрес.

Оба эти адреса могут применяться равноценно. Цифровой адрес удобен для обработки на компьютере, а доменный адрес - для восприятия пользователем.

Цифровой адрес имеет длину 32 бита. Для удобства он разделяется на четыре блока по 8 бит, которые можно записать в десятичном виде. Адрес содержит полную информацию, необходимую для идентификации компьютера.

Два блока определяют адрес сети, а два другие - адрес компьютера внутри этой сети. Существует определенное правило для установления границы между этими адресами. Поэтому IP - адрес включает в себя три компонента: адрес сети, адрес подсети, адрес компьютера в подсети.

Доменный адрес определяет область, определяющую ряд хост - компьютеров. В отличие от цифрового адреса он читается в обратном порядке. Вначале идет имя компьютера, затем имя сети, в которой он находится.

В системе адресов Internet приняты домены, представленные географическими регионами. Они имеют имя, состоящее из двух букв.

Пример Географические домены некоторых стран: Франция - fr; Канада - ca; США - us; Россия - ru.

Существуют и домены, разделенные по тематическим признакам. Такие домены имеют трехбуквенное сокращенное название.

Пример Учебные заведения - edu. Правительственные учреждения - gov. Коммерческие организации - com.

Службы Internet

1. Электронная почта (E-Mail).
2. Списки рассылки (MailList).
3. Телеконференции (Usenet).
4. Всемирная паутина WWW.
5. Служба передачи файлов (FTP).
6. Chat (разговор) в реальном времени (IRC).
7. ICQ.

Проблемы безопасности информации

Защита информации – контрольные механизмы, препятствующие незаконному использованию ресурсов.

Проведение финансовых операций с использованием Интернета или других сетей, заказ товаров и услуг, использование кредитных карточек, доступа к закрытым информационным ресурсам, передача телефонных разговоров требуют обеспечения соответствующего уровня безопасности. Начиная с 90-х годов, Интернет становится мощным средством обеспечения розничного торгового оборота, в нем циркулируют договорные и финансовые обязательства, необходимость защиты которых как от просмотра, так и от фальсификации, очевидна.

Конфиденциальная информация, которая передается по сети Интернет, проходит через определенное количество маршрутизаторов и серверов, прежде чем достигнуть пункта назначения. Существует возможность перехвата информации, более того, информация может быть изменена и перехвачена адресату в измененном виде.

Проблемы, возникающие с безопасностью передачи информации при работе в компьютерных сетях, можно разделить на следующие типы:

опасность заражения компьютерным вирусом;

перехват информации - целостность информации сохраняется, но ее конфиденциальность нарушена;

модификация информации - исходное сообщение изменяется либо полностью подменяется другим и отсылается адресату;

подмена авторства информации. Например, кто-то может послать письмо от вашего имени или Web-сервер может притворяться электронным магазином, принять заказы, номера кредитных карт, но не высылать никаких товаров.

Под термином "безопасность" подразумевается совокупность трех различных характеристик:

1. Аутентификация - процесс распознания пользователя системы и представления ему определенных прав и полномочий;

2. Целостность - идентичность отправленного и принятого;

3. Секретность - предотвращение несанкционированного доступа к информации.

Методы защиты информации

1. Антивирусная защита

Защиту от вирусов можно разделить на два класса: общие и профилактические средства. К общим средствам относятся - создание копий файлов и системных областей дисков и разграничение доступа - предотвращает использование компьютера без разрешения. Профилактические средства – наиболее обширный класс, к которому относятся антивирусные программы. Антивирусные программы разделяются на мониторы и сканеры. Монитор - находится резидентно (постоянно) в оперативной памяти компьютера и сообщает об обнаруженном вирусе. Сканер - программа, запускаемая пользователем (когда следует проверить дисковые накопители) или автоматически (периодическая проверка дисков). Важно, чтобы антивирусные программы, используемые для проверки, были самых последних версий. Широко используемые антивирусные программы - Drweb, AVP, Norton Antivir и др.

2. Криптография

Криптография - шифрование, позволяющая трансформировать данные в зашифрованную форму, из которой извлечь исходную информацию можно только при помощи ключа. Обеспечивает секретность информации.

В основе шифрования лежат два основных понятия: алгоритм и ключ. Алгоритм - это способ закодировать исходный текст, в результате чего получается зашифрованное послание. Зашифрованное послание можно интерпретировать только с помощью ключа.

Существуют две основные схемы шифрования: симметричное шифрование и несимметричное шифрование.

При симметричном шифровании отправитель и получатель владеют одним и тем же ключом, с помощью которого они могут зашифровать и расшифровать данные. Основной недостаток симметричного процесса заключается в том, что прежде чем начать обмен информацией, надо выполнить передачу ключа, а для этого опять-таки нужна защищенная связь, то есть проблема повторяется, хотя и на другом уровне. Если рассмотреть оплату клиентом товара или услуги с помощью кредитной карты, то получается, что торговая фирма должна создать по одному ключу для каждого своего клиента и каким-то образом передать им эти ключи. Это крайне неудобно.

Поэтому в настоящее время в Интернете используют несимметричное шифрование. В данной схеме для шифрования послания используется один ключ (от-

крытый), для расшифровки - другой (закрытый, личный). На самом деле это как бы две "половинки" одного целого ключа, связанные друг с другом.

Ключи устроены так, что сообщение, зашифрованное одной половинкой, можно расшифровать только другой половинкой (не той, которой оно было закодировано). Создав пару ключей, торговая компания широко распространяет публичный ключ (открытую половинку) и надежно сохраняет закрытый ключ (свою половинку).

Как публичный, так и закрытый ключ представляют собой некую кодовую последовательность. Публичный ключ компании может быть опубликован на ее сервере, откуда каждый желающий может его получить. если клиент хочет сделать фирме заказ, он возьмет ее публичный ключ и с его помощью закодирует свое сообщение о заказе и данные о своей кредитной карте. После кодирования это сообщение может прочесть только владелец закрытого ключа. Никто из участников цепочки, по которой передается информация, не в состоянии это сделать. Даже сам отправитель не может прочитать собственное послание, хотя ему хорошо известно содержание. Лишь получатель сможет прочесть сообщение, поскольку только у него есть закрытый ключ, дополняющий использованный публичный ключ.

Если фирме надо будет отправить клиенту квитанцию о том, что заказ принят к исполнению, она закодирует ее своим закрытым ключом. Клиент сможет прочитать квитанцию, воспользовавшись имеющимся у него публичным ключом данной фирмы. Он может быть уверен, что квитанцию ему отправила именно эта фирма, и никто другой, поскольку никто иной доступа к закрытому ключу фирмы не имеет.

Число бит в ключе определяет надежность защиты. Например, 8-битный ключ допускает 2^8=256 комбинаций ключей. Если использовать 128 - битный ключ, то необходимо будет перебрать 2^{128} ключей, что в настоящее время не под силу даже самым мощным компьютерам.

Даже если послание зашифровано, все равно остается возможность модификации исходного сообщения или подмены этого сообщения другим. Защита публичным ключом не является абсолютно надежной. Дело в том, что поскольку каждый желающий может получить и использовать чей-то публичный ключ, то он может сколь угодно подробно изучить алгоритм работы механизма шифрования и пытаться установить метод расшифровки сообщения, то есть реконструировать закрытый ключ.

Во многих странах вопрос применения алгоритмов шифровки данных находится в поле законодательного регулирования. В частности, в России к использованию в государственных и коммерческих организациях разрешены только те программные средства шифрования данных, которые прошли государственную сертификацию в административных органах, в частности, в Федеральном агентстве правительственной связи и информации при Президенте Российской Федерации.

3. Электронная подпись

Мы рассмотрели, как клиент может переслать организации свои конфиденциальные данные. Точно также он может общаться и с банком, отдавая ему рас-

поряжения о перечислении своих средств на счета других лиц и организаций. Однако здесь возникает проблема: как банк узнает, что распоряжение поступило именно от данного лица, а не от злоумышленника, выдающего себя за него? Одним из путей решения этой проблемы является передача получателю уникальной записи - электронной подписи.

Принцип ее создания тот же, что и рассмотренный выше. С помощью специальной программы (полученной от банка) создаются те же два ключа: закрытый и публичный. Публичный ключ передается банку. Если теперь надо отправить поручение банку на операцию с расчетным счетом, оно кодируется публичным ключом банка, а своя подпись под ним кодируется собственным закрытым ключом. Банк поступает наоборот. Он читает поручение с помощью своего закрытого ключа, а подпись - с помощью публичного ключа поручителя. Если подпись читаема, банк может быть уверен, что поручение ему отправили именно мы, и никто другой.

4. Сертификация Web-узлов

При заказе товаров в Интернете важно убедиться, что сервер, принимающий заказы и платежи от имени некоей фирмы, действительно представляет эту фирму. Подтвердить это может сертификат продавца. В сертификате указано, когда он выдан и на какой срок. Прежде чем выполнять платежи через Интернет или оправлять данные о своей кредитной карте кому-либо, следует проверить наличие действующего сертификата у получателя путем обращения в сертификационный отдел. Это называется сертификацией Web-узла. Например, в обозревателе Microsoft Internet Explorer 5.0, доступ к центрам сертификации осуществляется командой Сервис → Свойства Обозревателя → Содержание → Сертификатов → Доверенные Корневые Центры Сертификации.

5. Аутентификация

Аутентификация является одним из самых важных компонентов организации защити информации в сети. Прежде чем пользователю будет предоставлено право получить тот или иной ресурс, необходимо убедится, что он действительно тот, за кого себя выдает.

При получении запроса на использование ресурса от имени какого-либо пользователя сервер, представляющий данный ресурс, передает управление серверу аутентификации. После получения положительного ответа сервера аутентификации пользователю предоставляется запрашиваемый ресурс.

Одной из схем аутентификации является использование стандартных паролей. Эта схема является наиболее уязвимой с точки зрения безопасности - пароль может быть перехвачен и использован другим лицом. Чаще всего используются схемы с применением одноразовых паролей. Даже будучи перехваченным, этот пароль будет бесполезен при следующей регистрации, а получить следующий пароль из предыдущего является крайне трудной задачей.

V. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Задачей преподавателя при проведении лабораторных работ является грамотное и доступное разъяснение принципов и правил проведения работ, побуждение студентов к самостоятельной работе, определения места информатики в дальнейшей профессиональной работе будущего специалиста.

Цель лабораторной работы – научить студентов самостоятельно производить необходимые действия для достижения необходимого результата.

Прежде чем приступить к выполнению лабораторной работы, студенту необходимо ознакомиться с теоретическим материалом, соответствующим данной теме.

Выполнение лабораторной работы целесообразно разделить на несколько этапов:

- Формулировка и обоснование цели работы;
- Определение теоретического аппарата, применительного к данной теме;
- Выполнение заданий;
- Анализ результата;
- Выводы.

VI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ДОМАШНИХ ЗАДАНИЙ

Внеаудиторная работа по информатике включает в себя:

- Совершенствование и закрепление теоретических знаний, полученных на лекционных и лабораторных занятиях. Каждая тема курса включает вопросы входного контроля знаний (минимальный теоретический уровень), освоение которых необходимо для решения учебных задач, формирования умений и навыков темы.
- Формирование навыков практической работы - доведение умений до автоматизма путем решения упражнений - заданий, требующее повторного выполнения действий с целью его усвоения.
- Выполнение творческих работ, предусмотренных рабочей программой (см. пункт самостоятельная работа студентов).

При выполнении домашней работы студенты могут использовать различные источники приобретения информации: конспекты лекций, учебно-методические материалы курса, ссылки на научную литературу в информационном пространстве Интернета и др.

VII. КОМПЛЕКТ ЗАДАНИЙ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

Тема: Компьютерная графика.

Тема №1: Создание простейших изображений в графическом редакторе CorelDraw.

Задания к лабораторной работе:

В качестве отчетной работы требуется создать визитку с учетом основных правил подготовки изображений к печати в типографии и вывести на лист А4 при помощи автоматического расположения на листе.

1. Создание визитки

- Устанавливаем размер листа под размер визитки (90x50)
- Заполняем документ содержимым (в данном случае это несколько надписей и растровая картинка)
- Важно! Видимую рамку к визитке делать нельзя, т.к. из-за погрешностей при обрезке рамка будет иметь неодинаковую толщину или с каких-то сторон ее не будет вообще.
- Сохраняем документ.

В результате получаем:

Тема №2: Создание простейших изображений в графическом редакторе PhotoShop.

Задания к лабораторной работе:

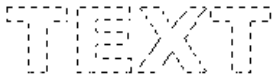
1. Создание контурного и обведенного текста.

Инструмент Type (Текст) обладает незначительными возможностями создания контуров. Невозможно задать толщину обводки, нельзя выбрать режим наложения или установить величину непрозрачности. Обычно для ввода и обводки текста перед импортом в Photoshop используют программу типа Illustrator. Если это невозможно, то для создания контуров воспользуйтесь командой Stroke (Выполнить обводку) меню Edit (Редактирование).

Прозрачный текст с цветным контуром можно создать, выполнив следующие действия:

1. Выберите инструмент Type Mask (Текст-маска) и щелкните на открытом изображении или на новом слое, в котором хотите поместить текст.

2. Введите текст. Оставьте выделенную область активной для выполнения последующих действий.



3. Задайте желаемый цвет контура.

4. В меню Edit (Редактирование) выберите команду Stroke (Выполнить обводку) и введите необходимые значения.

Для создания окрашенного текста, контур которого имеет другой цвет, выполните следующие действия:

1. Задайте основной цвет, соответствующий цвету текста.

2. Выберите инструмент Type (Текст), щелкните на новом слое и введите требуемый текст.

3. Выполните Select → Load Selection, чтобы выделить текст по контуру и склейте слои (Ctrl + E).

4. Задайте основной цвет, соответствующий цвету обводки.

5. В меню Edit (Редактирование) выберите команду Stroke (Выполнить обводку) и введите необходимые значения.

2. Пламя.

Нам потребуется однослойное изображение - белая надпись на черном фоне. Удобнее всего использовать режим серых полутонов. Эффект пламени более выразителен на широком светлом шрифте без засечек. Можно применить и массивный шрифт, обведя буквы по контуру.

Первый этап состоит в применении фильтра Wind, вызываемого через меню Filter → Stylize. Перед вызовом эффекта ветра надо повернуть изображение на 90° по часовой стрелке (команда Image → Rotate Canvas → 90° CW), поскольку в фильтре Wind нет вертикального направления ветра. Мы включим направление From the right, а силу ветра оставим минимальную - Wind. Чтобы следы ветра получились более мощными, можно применить фильтр несколько раз подряд, нажав клавиши Crtl + F. После чего возвращаем изображение в нормальное положение командой меню Image → Rotate Canvas → 90° CCW.

Придадим надписи и языкам ветра рваную форму, более соответствующую пламени. Для этого воспользуемся фильтром Diffuse, вызвав его через то же меню Filter → Stylize и установив режим Normal.

Далее применим фильтр Gaussian Blur при небольшом радиусе. В данном случае он составил 1.3 пиксела. Усилим дрожание с помощью фильтра Ripple (рябь), который вызывается через меню Filter → Distort. Зададим амплитуду (Amount) порядка 80 - 100, да и длина волны пусть будет небольшой, выберем Small или Medium.

На последнем этапе надо перевести изображение из черно-белого полутонового в индексированный цвет Image → Mode → Indexed Color. Изменить палитру изображения с индексированным цветом можно с помощью команды меню Image → Mode → Color Table. Редактор Photoshop предлагает несколько заранее подготовленных палитр, из которых выберем Black Body.

3. Лед.

Создадим новое изображение и напишем на нем текст черным цветом. Затем выполним Select → Load Selection и Layer → Flatten Image.

Выполним команду Select → Inverse, и применим фильтр Filter → Pixelate → Crystallize с коэффициентом кристаллизации равным 8.

Опять выполняем команду Select → Inverse, и применяем фильтр Filter → Noise → Add Noise с коэффициентом шума около 70 (включите Distribution: Gaussian).

Размываем избражение фильтром Filter → Blur → Gaussian Blur с коэффициентом размывки около 2. Затем отменяем выделение Select → None и используем фильтр Filter → Blur → Blur.

Выполняем команду Image → Adjust → Invert (Ctrl + I).

Повернем изображение Image → Rotate Canvas → 90 CW. Затем применяем фильтр Filter → Stylize → Wind (Method: Wind, Direction: From the left) и возвращаем картинку в горизонтальное положение Image → Rotate Canvas → 90 CCW.

Придадим изображению цвет льда с помощью команды Image → Adjust → HueSaturation (Включите Colorize, установите параметр Hue в положение - 181, а Saturation - 80).

Добавим искрящиеся блики с помощью дополнительных кистей. В меню Brushes (Кисти) выберите команду Load Brushes (Загрузить Кисти) и откройте файл assorted.abr из подкаталога Brushes. Установите текущим белый цвет и нанесите искорки на некоторые выступающие льдинки.

4. Золотая надпись.

Создаем новый белый лист. Переходим в палитру каналов и, нажав на кнопку внизу этой палитры, создаем канал #4. Его фон должен быть черным, а текст белым. Пишем текст. Этот эффект наиболее выразителен при использовании крупных символов.

Активизируем единственный слой Background он должен быть белым. Маску из канала #4 загружаем в окно редактора комбинацией клавиш [Ctrl]+[Alt]+[4]. Закрашиваем ее черным. Залить маску основным цветом можно клавишами [Alt]+[Del]. Выключаем маску, нажав [Ctrl]+[D].

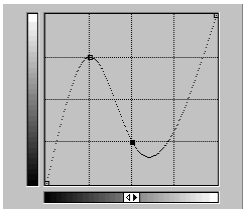
Надпись на слое Background надо растушевать с помощью гауссового фильтра (Filters → Blur → Gaussian Blur). Радиус в 3 пиксела.

Переименуем этот слой. Щелкните дважды мышкой по пиктограмме слоя Background в палитре слоев. Затем необходимо продублировать слой. Для этого

перетащите мышкой в палитре слоев дублируемый слой на кнопку создания нового.

Слои надо сдвинуть друг относительно друга (Filter → Other → Offset). Для одного из слоев набираем два положительных смещения на 4 пиксела, а для другого - два отрицательных по 4 пиксела.

Для верхнего из слоев (Layer 0 copy) режим наложения заменяем с Normal на Difference (разница). Теперь слои объединяем в один [Ctrl]+[E].



Нажав комбинацию клавиш [Ctrl]+[M], Вам надо попытаться воспроизвести кривую, подобную той, что на иллюстрации.

Загружаем маску #4 командой меню Select → Load Selection, затем маску инвертируем [Ctrl]+[Shift]+[I]. Убедившись, что маска включает в себя не надпись, а фон вокруг нее (об этом свидетельствует бегущая штриховая линия по краю окна редактора), нажмите клавишу [Del].

Нажимаем [Ctrl]+[D] и [Ctrl]+[I] - инвертирование. Создаем новый слой (цветной) Слой1. Подбираем ему из каталога золотистый цвет. Делаем заливку [Ctrl]+[Del]. Делаем для Слой0 цветность.

Надо объединить слои в срезаемую группу. Получаем ее, щелкнув мышкой при нажатой клавише [Alt] по линии, разделяющей слои в палитре Layers.

Создаем новый слой Слой2, делаем его цветным. В каталоге выбираем черный цвет и нажав [Ctrl]+[Del] заливаем. Расположите слои в такой последовательности: слой1, слой2, слой0.

5. Эффект «X-Files».

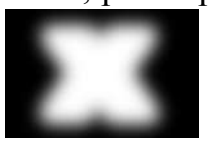
Создайте новое изображение RGB с чёрным фоном. Напишите ваш текст. Далее необходимо объединить слои (Ctrl + E). Теперь, откройте палитру Каналы (Window → Show Channels); переместите любой из каналов к значку новый канал. Это создаст новый канал. Откройте двойным щелчком мыши свойства этого канала и переименуйте его в «оригинал».

Перетащите канал «оригинал» к значку нового канала и сделайте дубликат. Дважды щелкните на новом канале, и переименуйте его в «белый». Затем нажав Ctrl + щелчок мыши на канале, загрузите его в выделение. Теперь мы немного расширим его. Используйте Select → Modify → Expand со значением 1 пиксела. Далее выполните Edit → Fill use: white. Снимите выделение Select → Deselect и выполните размывку Filter → Blur → Gaussian Blur с радиусом равным 2.

Перетащите канал «белый» к значку новый канал, чтобы у нас получился ещё один канал. Дважды щёлкните на образовавшемся канале, и переименуйте его в «жёлтый». Ctrl + щелчок, чтобы загрузить в выделение. Опять, Select → Modify → Expand со значением 2. Заполните белым цветом, снимите выделение и примените фильтр

Filter → Blur → Gaussian Blur с радиусом 3.

Последний канал мы получим следующим образом - перетащите канал «желтый» к значку новый канал. Переименуйте его в «зелёный», Ctrl + щелчок; расширьте на 4. Заполните белым, снимите выделение, размойте с радиусом 7 пикселов.



Переключитесь на канал RGB. Выполните Select → Load Selection и выберите там канал «зелёный». Теперь выберите подходящий зеленый цвет, например R:0 G:255 B:0. И выполните заливку Edit → Fill use: Foreground Color.

Сделайте тоже самое для канала «жёлтый», но с использованием цвета R:128 G:255 B:0. Теперь тоже самое сделайте для канала «белый», используя белый цвет. Загрузите «оригинал», выполните Select → Modify → Contract установка - 1 пиксел и залейте черным Edit → Fill use: black.

<u>Тема:</u> Операционная оболочка Total Commander.

<u>Задания к лабораторной работе</u>:

1. На диске C: создать два каталога DOG и GRAND.
2. В каталог DOG создайте файл dogovor.txt, содержащий текст:

   ДОГОВОР №1

   Данный договор составлен 10.10.2000 год.
3. Скопируйте файл dogovor.txt в каталог GRAND.
4. Переименовать в каталоге GRAND файл dogovor.txt на dogovor1.txt.
5. В файле dogovor1.txt исправить число 10.10.2000 на 10.12.2002.
6. Перенести файл dogovor1.txt в каталог DOG с именем itog.txt.
7. Просмотреть содержимое файла itog.txt.
8. Переименовать каталог DOG в DATA.
9. В корневом каталоге диска C: создать каталог с именем BLANK.
10. В каталоге BLANK создать файл doc1.txt, содержащий текст: Программа-оболочка – это программа, один из модулей которой, называемый резидентным, постоянно находится в оперативной памяти компьютера.
11. Скопировать файл doc1.txt в файл doc2.txt каталога BLANK.
12. Переименовать в каталоге BLANK файл doc2.txt в файл otchet.txt.
13. Добавить в файл otchet.txt следующий текст: Оболочки позволяет эффективно работать с файловой системой дисков, а также запускать программы на исполнение.
14. В корневом каталоге диска C: создать каталог KONTORA.
15. Перенести каталог BLANK в каталог KONTORA.
16. Скопировать одновременно все файлы из каталога DOG в BLANK.
17. В каталоге BLANK поместить файлы в архивный файл paper.rar одновременно уничтожив исходные файлы.
18. Извлечь файлы из архива paper.rar в каталог DATA.
19. Осуществить поиск файлов doc2.txt и dogovor1.txt.

<u>Тема:</u> Операционная система Windows.

Тема №1: Рабочий стол Windows. Работа с окнами

Задания к лабораторной работе:

1. Познакомьтесь с элементами Рабочего стола: значками объектов Windows и Панелью задач.

2. Отработка приемов работы с мышью:

– Выделить значок Мой компьютер.
– Переместить значок в новое место Рабочего стола (Если перетаскивание не удается, то вызвать контекстное меню Рабочего стола отменить режим «Упорядочивать» – «Автоматически»).
– Вернуть значок на место.
– Снять выделение со значка Мой компьютер.
– Открыть Главное меню кнопкой Пуск.
– Определить текущую дату, применив прием зависания к индикатору часов на Панели задач.
– Определить установленную раскладку клавиатуры, вызвав контекстное меню индикатора языка на Панели задач и выбрав в меню пункт Свойства.

3. На Рабочем столе создать с помощью мастера ярлыки для приложений MS Word и MS Excel.

4. Знакомство с элементами окна:

– Открыть окно папки «Мой компьютер» и познакомиться с элементами окна.
– Развернуть окно на весь экран; восстановить размер окна; свернуть окно; развернуть окно; закрыть окно.
– Открыть окна папок «Мой компьютер» и «Корзина».
– Изменить размеры окон таким образом, чтобы они не перекрывали друг друга.
– Закрыть окна.

5. Установить шлейф для мыши.

6. Включить заставку: ОБЪЕМНЫЙ ТЕКСТ, текст «Безопасность», параметры: стиль движения «Волны», шрифт – Arial.

7. Просмотреть предлагаемые рисунки и узоры Рабочего стола и установить наиболее понравившийся.

8. Установить новые свойства у Панели задач:

– раскрыть окно свойств Панели задач вызвав ее контекстное меню и выбрав в меню пункт Свойства;
– снять флажок отображения часов;
– установить флажок Автоматически убирать с экрана;
– проверить установленные свойства;
– восстановить прежнее значение измененных свойств.

9. Изучить свойства клавиатуры:

– в Панели управления открыть окно Клавиатура;
– установить комбинацию клавиш Ctrl+Shift для переключения между русским и английским языками;
– отключить индикатор раскладки клавиатуры;
– проверить установленные свойства;

– восстановить прежние значения свойств.

10. Изучить свойства Корзины:

– раскрыть окно Свойства корзины;
– выбрать закладку Глобальные;
– изменить значения свойства Уничтожать файлы сразу, не помещая их в корзину;
– изменить значения свойства Запрашивать подтверждение на удаления;
– проверить установленные свойства;
– восстановить прежнее значение измененных свойств.

11. Очистить список документов в Главном меню.

12. Создать в Главном меню новый пункт, запускающий программу Блокнот.

13. Удалить из Главного меню пункт Блокнот.

Тема №2: Стандартные программы. Работа с папками и файлами

Задания к лабораторной работе:

1. Записать в тетради перечень стандартных программ Windows.

2. Записать в тетради назначение программы «*Блокнот*».

3. Запустить программу «*Блокнот*».

4. Ввести текст: «Текст вводится с помощью буквенно-цифровых клавиш. Для ввода прописных букв используется одновременное нажатие клавиши Shift для ввода длинной последовательности прописных букв клавиатуру можно переключить с помощью клавиши Caps Lock. Для переключения между русскими и английскими символами используется индикатор языка на Панели задач, или специальная комбинация клавиш, установленная на Вашем компьютере. Обычно это комбинация клавиш Alt+Shift или Ctrl+Shift.»

5. Свернуть окно программы «*Блокнот*» на *Панель задач*.

6. Записать в тетради назначение программы *WordPad*.

7. Запустить программу *WordPad*.

8. Ввести текст: «Для выделения фрагмента текста необходимо нажать левую кнопку мыши на начале фрагмента и, удерживая ее, переместить указатель мыши в конец выделяемого фрагмента. Необходимый фрагмент выделится черным цветом».

9. С набранным текстом проделать следующую работу:

– Вставить набранный текст в конец документа два раза;
– Выделить первый абзац и задать для него следующие параметры: шрифт – Arial, размер шрифта – 16, цвет шрифта – зеленый, тип шрифта – полужирный курсив;
– Выровнять первый абзац по левому краю, второй абзац – по центру, третий абзац – по правому краю.

10. Свернуть окно программы *WordPad* на *Панель задач*.

11. Записать в тетради назначение программы *Paint*.

12. Запустить программу *Paint*.

13. Нарисовать следующий рисунок

14. Свернуть окно программы *Paint* на *Панель задач*.

15. Открыть папку «*Мой компьютер*» и перечислить находящиеся в ней объекты.

16. Изменить несколько раз размеры и расположение значков файлов и папок.

17. На диске *C:* создать папки *Деятельность, Безопасность, Все*.

18. В папке *Деятельность* создать папки *Экология, Энергия, Пожар*.

19. Развернуть программу «*Блокнот*» и сохранить текст с именем *Записки* в папке *Экология*.

20. Развернуть программу «*WordPad*» и сохранить текст с именем *Тезис* в папке *Энергия*.

21. Развернуть программу «*Paint*» и сохранить рисунок с именем *Разбор* в папке *Пожар*.

22. Скопировать папку *Деятельность* в папку *Безопасность*.

23. Скопировать папку *Безопасность* на диск *A:*.

24. Переместить на диске *C:* папку *Безопасность* в папку *Деятельность*.

25. Переименовать на диске *C:* папку *Безопасность* на *Все*.

26. Удалить папку *Все*.

27. Восстановить папку *Все*.

28. Выделить папку *Все* и посмотреть, сколько места на диске она занимает. Заархивировать папку *Все* в архив *Общие.rar*.

29. Сравнить занимаемое место на диске архивом *Общие.rar* с первоначальным объемом папки *Все*. На сколько уменьшился объем занимаемого места на диске?

30. Разархивировать архив *Общие.rar*.

Тема №3: Программа «Поиск». Программа «Проводник»

Задания к лабораторной работе:

1. Организовать поиск файлов по *Имени и размещению*:

- найти на диске *C:* все файлы с расширением *.tmp* (*.tmp);
- найти на диске *C:* все файлы с расширением *.bmp* (*.bmp);
- найти все файлы с однобуквенными именами и расширениями (?.?);

2. Организовать поиск файлов по *Дате* изменения:

- найти на диске *C:* файлы, созданные с 1.10.2000 по сегодняшний день;
- найти на диске *C:* файлы, созданные с 1.10.2004 по 15.07.2005;
- найти на диске *C:* файлы, созданные за последний месяц; за последние два месяца; за последний день; за последние три дня;

3. Используя вкладку *Дополнительно* найти файлы типа *Точечный рисунок BMP*. На каком диске вы провели поиск? Полученный список файлов представить в виде таблицы и отсортировать его по размеру. Просмотреть самый большой рисунок.

4. Найти на диске *C:* все текстовые файлы, содержащие слово «*Windows*».

5. Найти на диске *C:* все *Текстовые документы,* содержащие слово «*windows*», но не «*Windows*». Для этого необходимо отменить режим *Параметры → С учетом регистра*. Открыть один из найденных файлов и найти в нем указанное слово.

6. Запустить программу «*Проводник*».

7. Измените *Вид* правой панели с помощью меню.

8. Свернуть все папки, щелкая по значкам « – » на дереве папок в левой панели программы.

9. Показать в правой части содержимое диска *C:* и при помощи контекстного меню создать на диске *C:* папку *Европа*.

10. Развернуть в правой части диск *C:* чтобы увидеть созданную папку.

11. Открыть папку *Европа* в правой части *Проводника*.

12. Создать в папке *Европа* папку *Россия*.

13. В папке *Россия* создать текстовый документ *Записки1* и точечный рисунок *Записки2*.

14. Правой кнопкой мыши переместить их в папку *Европа*.

15. Скопировать файл *Записки2* в папку *Россия*.

Тема: Текстовый процессор Word.

Тема №1: Создание текстового документа.

Задания к лабораторной работе:

1. Запустить текстовый процессор MS Word.

2. Изучить структуру окна редактора и назначение его основных элементов (строка заголовка; кнопки управления окном *Свернуть*, *Развернуть/Восстановить, Закрыть*; строка меню; панели инструментов; линейка; рабочая область; полосы прокрутки; строка состояния; рамка).

3. Набрать следующий текст с соблюдением вышеуказанных правил:

Человек дистанционно не может определить находится ли установка под напряжением или нет.

Ток, который протекает через тело человека, действует на организм не только в местах контакта и по пути протекания тока, но и на такие системы как кровеносная, дыхательная и сердечно-сосудистая.

Возможность получения электрических травм имеет место не только при прикосновении, но и через напряжение шага и через электрическую дугу.

Электрический ток, проходя через тело человека оказывает термическое воздействие, которое приводит к отекам (от покраснения, до обугливания), электролитическое (химическое), механическое, которое может привести к разрыву тканей и мышц; поэтому все электрические травмы делятся на местные и общие (электроудары).

4. Сохранить документ с именем «Травмы».

5. Завершить работу с MS Word.

Тема №2: Форматирование текстов.

Задания к лабораторной работе:

1. Загрузить MS Word.

2. Установить поля станицы: верхнее – 2 см, нижнее – 2 см, левое – 3 см, правое – 1 см.

3. Напечатать следующий текст без форматирования символов:

Чрезвычайные ситуации.

Чрезвычайные ситуации (ЧС) – внешне неожиданная, внезапно возникающая обстановка, которая характеризуется резким нарушением установившегося процесса, оказывающая значительное отрицательное влияние на жизнедеятельность людей, функционирование экономики, социальную сферу и окружающую среду.

Ликвидация последствий ЧС осуществляется силами и средствами организаций местного самоуправления, на территории которых сложилась чрезвычайная ситуация, под непосредственным руководством соответствующей комиссии по ЧС. Если масштабы ЧС таковы, что имеющимися силами и средствами локализовать её невозможно, указанные комиссии обращаются за помощью к вышестоящей комиссии по ЧС.

Работы, связанные со спасением людей, проводятся непрерывно до полного их завершения. При необходимости организуется смена и отдых личного состава формирований на месте работ или в установленных местах (районах).

Целью проведения аварийно-спасательных работ и других неотложных работ (АСДНР) является спасение людей и оказание медицинской помощи пострадавшим, локализация аварии и устранение повреждений, препятствующих ведению спасательных работ.

Спасательные работы включают*:* во-первых, разведку маршрутов выдвижения формирований и участков (объектов) работ; во-вторых, локализацию и тушение пожаров на маршрутах выдвижения и участках (объектах) работ; в-третьих, розыск поражённых и извлечение их из повреждённых и горящих зданий, загазованных, затоплённых и задымлённых помещений, завалов; в-четвертых, вскрытие разрушённых, повреждённых и заваленных защитных сооружений и спасение находящихся людей.

Понятие чрезвычайные ситуации (ЧС) в соответствии с текстом Федерального закона «О защите населения и территорий от чрезвычайных ситуаций природного и техногенного характера» можно сформулировать как неблагоприятную обстановку на определённой территории, сложившуюся в результате аварии, катастрофы или иного бедствия, которые могут повлечь или повлекли за собой человеческие жертвы, ущерб здоровью людей, окружающей среде, значительные материальные потери и нарушения жизнедеятельности людей.

4. Отформатировать 1 абзац: шрифт – Arial, размер – 24, видоизменение слов «Чрезвычайные ситуации » – К, интервал после – 16 пт, межстрочный интервал – одинарный, выравнивание – по центру, цвет шрифта – красный, перед абзацем вставить символ - ✍.

5. Отформатировать 2 абзац: шрифт – Book Antiqua, размер – 15, видоизменение слов «Ликвидация последствий» – подчеркнутый, отступ слева – 1,5 см, отступ справа – 1 см, отступ первой строки – 1,25 см, интервал перед – 3 пт, интервал после – 8 пт, межстрочный интервал – полуторный, выравнивание – по левому краю, цвет шрифта – зеленый.

6. Отформатировать 3 абзац: шрифт – Monotype Corsiva, размер - 20, видоизменение слов «со спасением людей» – К, отступ слева – 1 см, отступ справа – 1 см, отступ первой строки – 1,27 см, межстрочный интервал – двойной, выравнивание – по правому краю, цвет шрифта – синий, разбить на две колонки.

7. Отформатировать 4 абзац: видоизменение слов «аварийно-спасательных работ» – К.

8. В пятом абзаце разбить нумерацию (во-первых, во-вторых и т.д.) на отдельные строки.

9. Отформатировать 6 абзац: шрифт – Times New Roman, размер – 16, видоизменение слов «Федерального закона» – Ж, отступ слева – 0 см, отступ справа – 0 см, отступ первой строки – 1,5 см, межстрочный интервал – двойной, выравнивание – по ширине, цвет шрифта – синий.

10. Отформатировать заголовок документа: начертание – ЖК, отступ перед – Авто, отступ после – Авто, размер – 30, цвет – желтый.

11. Скопировать первые два абзаца в конец документа и разбить скопированный текст следующим образом: одно предложение – один абзац.

12. Автоматически пронумеровать страницы документа: номер на первой странице не ставить, выравнивание от центра, внизу.

13. Вставить в документ колонтитул: верхний колонтитул – ФИО, выравнивание – по правому краю, нижний – автоматическая вставка даты создания документа, выравнивание – по центру.

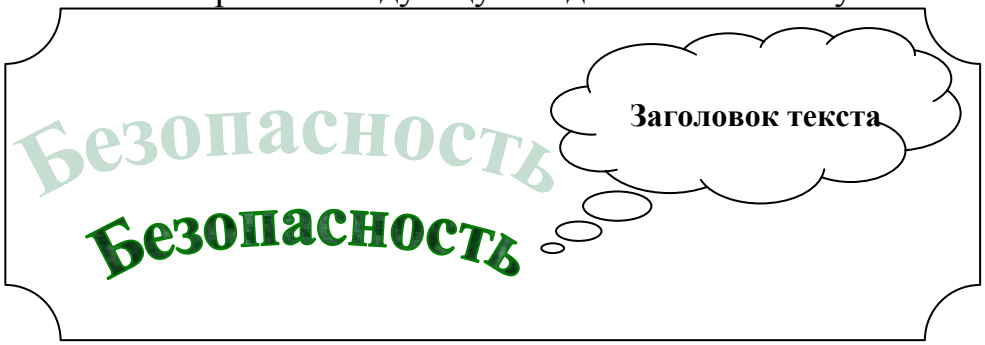
14. Проверить документ на наличие орфографических ошибок.

15. Установить автоматическую расстановку переносов.

16. Вставить из *Коллекции картинок* любую из понравившихся.

17. Изобразить следующий заголовок с помощью инструмента Word Art:

17. Изобразить следующую надпись к заголовку:



18. Отредактировать рисунок следующим образом:

19. Набрать следующий текст:

Горение — химическая реакция, которая сопровождается выделением тепла и света. Для осуществления горения необходимо: окислитель (кислород); источник возгорания; источник пламени. Если речь идёт о горючих веществах, то степень пожарной опасности горючих веществ характеризуется: температурой вспышки; температурой воспламенения; температурой самовоспламенением. Температура вспышки – минимальная температура, при которой над поверхностью жидкости образуется смесь паров этой жидкости с воздухом, способная гореть при поднесении открытого источника огня. Температура воспламенения – минимальная температура, при которой вещество загорается от открытого источника огня и продолжает гореть после его удаления. Температура самовоспламенения – минимальная температура, при которой происходит его воспламенение на воздухе за счет тепла химической реакции без поднесения открытого источника огня.

20. Установить для первой буквы первого слова абзаца буквицу: шрифт – Arial, положение – в тексте, высота в строках – 5.

21. Вставить перед текстом (задание 7) заголовок «Пожарная безопасность» и установить для него эффект «Мигающий фон».

22. Сохранить документ с именем «Пожар».

Тема №3: Создание таблиц и схем.

Задания к лабораторной работе:

1. Вставить таблицу любым из перечисленных способов, содержащую 8 строк и 6 столбцов.

2. Объединить ячейки следующим образом:

3. Вставить перед таблицей заголовок «Согласные звуки русского языка»: шрифт – Monotype Corsiva, размер – 16, начертание - Полужирный курсив.

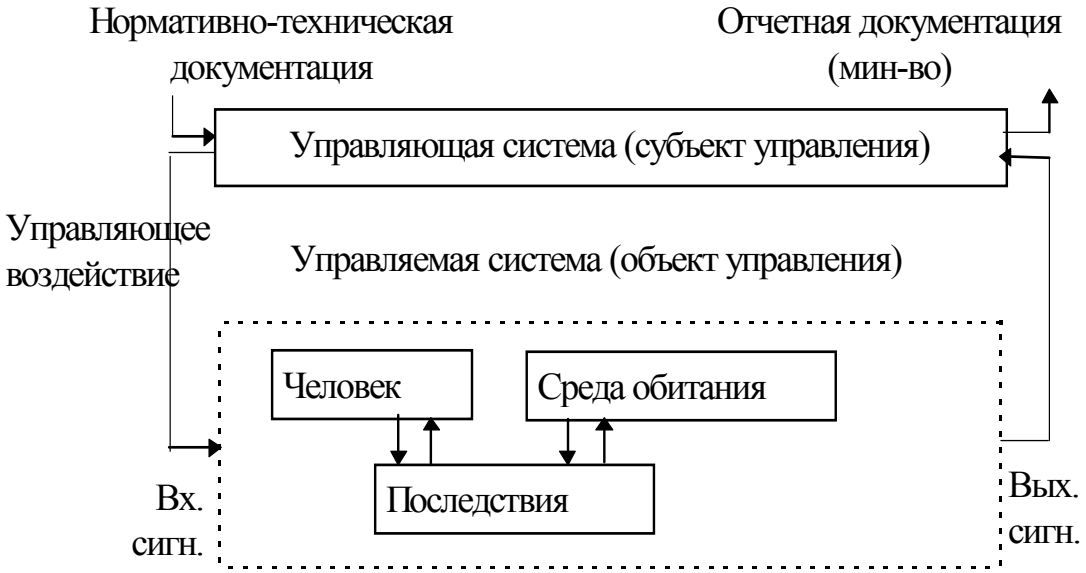
4. Заполнить таблицу следующим образом. Обратите внимание на выравнивание текста в ячейках таблицы.

Вероятность воздействия		Вероятность обнаружения		Серьезность последствий	
критерии	категории	критерии	категории	критерии	категории
очень высокая	5	обнаруживается	0	нет	0
высокая	4	очень высокая	1	малая	2
средняя	3	высокая	2	низкая	4
низкая	2	средняя	3	средняя	6
очень низкая	1	низкая	4	высокая	8
нет	0	очень низкая	5	очень высокая	10

5. Оформить таблицу следующим образом.

- Внешняя граница – синий пунктир.
- Внутренняя граница – сплошная синяя.
- Заливка первой строки – оранжевая.
- Заливка столбцов «критерии» - желтая.

6. Изобразить следующий объект. Самостоятельно задать цветовую гамму его элементам.



Тема №4: Списки, стили, оглавления.

Задания к лабораторной работе:

1. Набрать следующий перечень (каждое слово на новой строке): существительное, прилагательное, наречие, глагол. Выделить текст и оформить его в виде *Нумерованного списка* с помощью кнопки .

2. Набрать следующий перечень (каждое слово на новой строке): Пушкин, Лермонтов, Некрасов, Толстой. Выделить текст и оформите его в виде *Маркированного списка* с помощью кнопки .

3. Изменить нумерованный список: нумерация начинается с 5, нумерация – один, два, ….

4. Изменить маркированный список: знак маркера – .

5. Набрать следующий текст без соблюдения начертания (каждая фраза с новой строки): Введение в информатику, **Краткая история развития информационных технологий,** Введение, **Информация,** Понятие информации, Информационные системы, Системы счисления, *Понятие системы счисления, Непозиционные системы счисления, Позиционные системы счисления,* Представление информации в компьютере, *Общие сведения, Формы и коды представления данных в памяти ЭВМ,* **Технические средства информационных технологий,** Компьютер, Базовая структура персонального компьютера, Устройства ввода-вывода, **Программное обеспечение компьютера, Компьютерные сети,** Локальные компьютерные сети, Глобальные компьютерные сети.

6. Создать следующий многоуровневый список:

– *Уровень 1*: Формат номера – Глава, нумерация – 1, 2, 3 ..., начать с – 1; Положение номера: Выравнивание – по левому краю; Отступ – 0,5 см; От номера до текста – 0 см. Шрифт – Полужирный.
– *Уровень 2*: Формат номера – 1.1., нумерация – 1, 2, 3 ..., начать с – 1; Положение номера: Выравнивание – по левому краю; Отступ – 1 см; Шрифт – Обычный.
– *Уровень 3*: Формат номера – 1.1.1., нумерация – 1, 2, 3 ..., начать с – 1; Положение номера: Выравнивание – по левому краю; Отступ – 1,5 см; Шрифт – Курсив.

7. Применить к набранному тексту созданный список следующим образом: первая трока – заголовок текста (выравнивание – От центра, начертание – Полужирный курсив); текст, выделенный жирным шрифтом – Уровень 1; подчеркнутый текст – Уровень 2; текст, выделенный курсивом – Уровень 3.

8. Набрать следующий текст:

А. Пушкин
Тиха украинская ночь. Прозрачно небо,
Звезды блещут.
Своей дремоты превозмочь
Не хочет воздух.
Я. Некрасов
Быстро лечу я по рельсам чугунным,
Думаю думу свою.
Б. Пастернак
Мело, мело по всей земле во все пределы.
Свеча горела на столе, свеча горела.
М. Лермонтов
Горные вершины спят во тьме ночной.
Тихие долины полны свежей мглой.
С. Маршак
Вокруг белеющих прудов
Кусты в пушистых полушубках,
И проволока проводов
Таится в белоснежных трубках.

9. Применить стиль *Заголовок1* к фамилиям А. Пушкин, Я. Некрасов, Б. Пастернак.

10. Создать новый стиль: имя – Поэт, основа на стиле – Заголовок1, стиль следующего абзаца – обычный. Форматирование: шрифт – Monotype Corsiva, размер – 14, начертание - Полужирный курсив, выравнивание – От центра, межстрочный интервал – двойной.

11. Применить стиль *Поэт* к фамилиям М. Лермонтов, С. Маршак.

12. Разбить текст на страницы таким образом, чтобы каждый автор начинался на новой странице.

13. На последней странице вставить оглавление.

14. Сохранить документ с именем «Стили».

Тема №5: Язык и статистика: редактирование формул.

Задания к лабораторной работе:

1. Вставить в документ формулы согласно варианту, указанному преподавателем.

Вариант 1.

$$\int_0^t \frac{dQ}{Q^4+\frac{\mathrm{Bi}}{\mathrm{Sk}}Q-\left(1+\frac{\mathrm{Bi}}{\mathrm{Sk}}\right)}=\frac{\alpha_1+2\alpha_0}{\left(1-\alpha_0+\frac{\alpha_1}{2}\right)\sqrt{\alpha_1^2+\sigma\alpha_0^2}};$$

$$\begin{cases} a_1\sum_{i=1}^{n}x_i+a_0 n=\sum_{i=1}^{n}y_i \\ a_1\sum_{i=1}^{n}x_i^2+a_0\sum_{i=1}^{n}x_i=\sum_{i=1}^{n}x_i y_i \end{cases};$$

$$\begin{pmatrix} \sin\lambda_1 & 0 & 0 \\ 0 & \cos\lambda_2 & 0 \\ 0 & 0 & \sin\lambda_3 \end{pmatrix}.$$

Вариант 2.

$$c_{эф}\rho\Delta z\frac{t_{i,\,k}^{n+\frac{1}{2}}-t_{i,\,k}^{\mathrm{n}}}{\Delta\tau}=\frac{\alpha}{\left(\frac{h_0}{2}+\frac{\Delta Z}{2}\right)}\sum_{i=1}^{M}\left(t_{i,\,k-1}^{n}-t_{i,\,k}^{\mathrm{n}}\right);$$

$$\begin{cases} 4x^3-4x+4y=0 \\ 4y^3+4x-4y=0 \end{cases};$$

$$\Delta=-\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}.$$

Вариант 3.

$$\int_{\tau_1}^{\tau_k}\frac{\chi}{\varphi_2(\tau)}e^{\frac{-\beta}{\varphi_2(\tau)}}d\tau=\int_{\tau_1}^{\tau_k}\frac{\chi}{x_2(\tau)}e^{\frac{-\beta}{x_2(\tau)}}d\tau+\frac{B}{1+\frac{Bi}{Sk}};$$

$$\begin{cases} a_2\sum_{i=1}^{n}x_i^2+a_1\sum_{i=1}^{n}x_i+a_0=\sum_{i=1}^{n}y_i \\ a_2\sum_{i=1}^{n}x_i^3+a_1\sum_{i=1}^{n}x_i^2+a_0\sum_{i=1}^{n}x_i=\sum_{i=1}^{n}x_i y_i \\ a_2\sum_{i=1}^{n}x_i^4+a_1\sum_{i=1}^{n}x_i^3+a_0\sum_{i=1}^{n}x_i^2=\sum_{i=1}^{n}x_i^2 y_i \end{cases};$$

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}.$$

Вариант 4.

$$S=\left(i_g+0{,}5\right)C_{эф}\rho\Delta R\frac{\sum_{j=1}^{N}\left(t_{i+1,\,j}^{n+1}-t_{i+1,\,j}^{n+\frac{1}{2}}\right)}{\Delta\tau}+\sqrt{\frac{Q-b}{1-Q}};$$

$$\begin{cases} \frac{\partial\Phi}{\partial x}=-5+2x\lambda=0 \\ \frac{\partial\Phi}{\partial y}=-7+2y\lambda=0 \\ x^2+y^2=16 \end{cases};$$

$$\left(\begin{array}{cccc|ccccc} a_{11} & a_{12} & \dots & a_{1n} & 1 & 0 & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & a_{2n} & 0 & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & 0 & 0 & 0 & \dots & 1 \end{array}\right).$$

Вариант 5.

Вариант 6.

$$t_{ср} = \frac{\rho_{ж}\Delta R^2 \Delta Z \cdot 6{,}28\sum_{i=1}^{N}\sum_{j=1}^{M}\left(t_{i,j}^{r} + A\right)}{M_{\sigma}^{n+1}} + \sqrt{at};$$

$$\frac{dT(F_0)}{dF_0} = K\left\{\frac{\frac{\alpha}{\lambda}\left[T_c(F_0) - T(F_0)\right]}{\sum_{i=1}^{n}(K_i - K_0)} + \sqrt{\frac{\sigma_B R}{\lambda}}\right\};$$

$$\begin{cases} x_1 = 1 + \sin\dfrac{20}{\sqrt{74}}t \\ y_1 = 1 - \cos\dfrac{28}{\sqrt{74}}t \\ z_1 = 1 + \sin\dfrac{\sqrt{74}}{8}t \cdot \cos\dfrac{\sqrt{74}}{8}t \end{cases};$$

$$\begin{cases} \dfrac{\partial z}{\partial x} = xy(8 - 3x - 2y) = 0 \\ \dfrac{\partial z}{\partial y} = x^2(4 - x - 2y) = 0 \end{cases};$$

$$\begin{vmatrix} 0-\lambda & 0{,}2 & 1 \\ 1 & 0-\lambda & 0 \\ 0 & 0{,}8 & 0-\lambda \end{vmatrix} = 0.$$

$$\begin{pmatrix} -2 \\ 17 \\ 5 \end{pmatrix} = \alpha_1\begin{pmatrix} 1 \\ 3 \\ 6 \end{pmatrix} + \alpha_2\begin{pmatrix} -3 \\ 4 \\ 5 \end{pmatrix} + \alpha_3\begin{pmatrix} 1 \\ -7 \\ 2 \end{pmatrix}.$$

Тема: Электронная таблица Excel

Тема №1: Знакомство с электронной таблицей.

Задания к лабораторной работе:

1. Запустить табличный процессор MS Excel.
2. Выделить:
   - Строки 5, 6, 9, 13. Снять выделение.
   - Столбцы B, C, F, G, K. Снять выделение.
   - Совокупность строк 1, 3, 6, 7, 10, 11 и столбцов B, D, G, H, I. Снять выделение.
   - Блок ячеек B5: F10.
   - Совокупность блоков A1:C4, B6:E12, G8:H10.
3. Вставьте два новых листа перед рабочим листом *Лист3*, используя оба способа.
4. Переименовать *Лист1* в *Глагол*; *Лист2* в *Наречие*.
5. Поменять местами листы *Глагол* и *Наречие*.
6. Создать копии рабочих листов *Глагол* и *Наречие*.
7. Удалить копии листов *Глагол* и *Наречие.*
8. Сохранить рабочую книгу с именем «Части речи».

Тема №2: Создание простейшей таблицы. Ввод и редактирование данных. Формат данных.

Задания к лабораторной работе:

1. В ячейку *A1* внести слово *Литература*.
2. Создайте ниже приведенную таблицу.

№ п/п	ФИО	Дата рождения	Век
	Пушкин А.С.	6.06.1799	
	Гоголь Н.Ф.	1.04.1809	
	Лермонтов М.Ю.	3.10.1814	
	Толстой Л.Н.	28.08.1828	
	Блок А.А.	28.11.1880	

	Маяковский В.В.	7.07.1893	
	Чернышевский Н.Г.	24.07.1829	
	Горький А.М.	16.03.1868	
	Достоевский Ф.М.	11.11.1821	

3. С помощью автозаполнения заполнить первый столбец таблицы таким образом: 1 автор, 2 автор, 3 автор и т.д.

4. Самостоятельно определить век, в котором родился каждый автор, и заполнить соответствующий столбец.

5. Отредактировать в ячейке *A1* слово *Литература* на *Русская литература*.

6. После четвертой строки вставить пустую строку и заполнить ее таким образом: ФИО – Цветаева М.И., Дата рождения – 26.09.1892.

7. Вставить между столбцами *ФИО* и *Год рождения* новый столбец – *Произведение* и заполнить его самостоятельно.

8. В столбце *Дата рождения* изменить формат представления информации на *Дата*, обозначение – 6 марта 1982 г (пример).

9. Сохранить документ с именем «Таблица».

Тема №3: Адресация. Вычисления в таблицах.

Задания к лабораторной работе:

1. В одном из опытов изучались частоты частей речи в прозе К. Федина. Было взято 10 выборок по 500 знаменательных слов каждая. В выборки включалась только авторская художественная речь. Были получены следующие выборочные частоты имен существительных и имен прилагательных:

Проза К.Федина

Части речи	**Частоты выборок (x_i)**										**Ср. частота**
	1	2	3	4	5	6	7	8	9	0	
ествительное	82	87	18	73	58	01	22	33	13	94	
агательное	9	1	3	0	3	3	2	9	9	1	
ол	5	7	4	04	00	12	3	1	05	08	

- На *Листе1* создать приведенную выше таблицу.
- Переименовать *Лист1* в *Средняя*.
- Найти среднюю частоту выборки для каждой части речи по формуле $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$, где n – количество выборок (наблюдений). Сделать вывод.

2. Переименовать *Лист2* в *Отклонение*.

3. На листе *Отклонение* создать приведенную ниже таблицу и найти отклонение выборочных частот от средней частоты σ для каждой части речи по формуле $\sigma = \sqrt{\frac{\sum_{i=1}^{n} a_i^2}{n}}$, где $a_i = x_i - \bar{x}$.

Выборки	**Выборочные частоты, их отклонение от средней частоты и квадраты этих отклонений**								
	Существительное			*Прилагательное*			*Глагол*		
	x_i	a_i	a_i^2	x_i	a_i	a_i^2	x_i	a_i	a_i^2

1-ая									
2-ая									
…									
Сумма									
$\bar{x}$									
σ									

4. Переименовать *Лист3* в *Предложения*.

5. На листе *Предложения* создать таблицу сравнения количества предложений текста некоторого объема у различных авторов.

Автор	**Количество предложений**	**Сравнение**
Толстой Л.Н.	358	
Чернышевский Н.Г.	226	
Макаренко А.С.	345	
Новиков Н.И.	289	
Одоевский В.Ф.	456	
Горький А.М.	384	
Общее количество		
Среднее количество		
Максимальное количество		
Минимальное количество		

– Найти общее количество предложений.

– Найти минимальное количество предложений по всем авторам.

– Найти максимальное количество предложений по всем авторам.

– Найти среднее количество предложений по всем авторам.

– Используя функцию ЕСЛИ определить «выше» или «ниже» среднего находится количество предложений текста каждого автора.

6. Сохранить документ с именем «Язык и статистика».

Тема №4: Сортировка данных и форматирование таблиц.

Задания к лабораторной работе:

1. Запустить MS Excel.
2. Открыть файл *Язык и статистика.xls.*
3. К таблице на листе *Средняя* применить следующие атрибуты форматирования:
- выравнивание;
- шрифты;
- цвет фона;
- ширина столбцов и высота строк;
- рамка.
5. К таблице на листе *Отклонение* применить стиль автоформатирования – *Классический 2*.
6. На листе *Предложения* создать три копии таблицы (на этом же листе).

7. К каждой таблице на листе *Предложения* применить стили автоформатирования – *Цветной 1*, *Цветной* 2, *Цветной 3*.
8. На листе *Предложения* отсортировать первую таблицу – столбцу Количество предложений, вторую таблицу – по столбцу Автор.
9. Сохранить рабочую книгу с именем «Язык и статистика (формат)».

Тема №5: Графический анализ данных.

Задания к лабораторной работе:

1. По имеющимся данным: «за» – 7 чел., «против» – 11 чел., «воздержалось» – 2 чел., построить круговую объемную диаграмму, отражающую процентное соотношение результатов голосования. Цвета секторов – красный, синий и зеленый соответственно.

2. По данным таблицы построить различные типы диаграмм, отображающих:

– долю каждого наименования в первом предложении;
– динамику изменения количество подлежащих во всех предложениях;
– сравнение общего числа наименований для всех предложений;
– сравнение количества сказуемых в первом, третьем и пятом предложениях.

Наименование	**1 предложение**	**2 предложение**	**3 предложение**	**4 предложение**	**5 предложение**	**6 предложение**
Подлежащее	11	12	12	15	17	19
Сказуемое	15	21	20	20	17	19
Деепричастие	23	29	32	34	30	33
Итого						

3. Создать следующую таблицу и выполнить к ней задания.

Факультет Группа	1 курс	2 курс	3 курс	4 курс	5 курс	Макс. кол-во	Миним. кол-во	Среднее
ФПИ	75	68	68	62	56			
ФМО	60	58	55	57	49			
ФФ	100	91	92	85	69			

– вычислить максимальное, минимальное и среднее количество студентов по отдельным факультетам;
– построить диаграмму, отображающую сравнение количества студентов на разных курсах по факультету ФФ (круговая или кольцевая); разместить диаграмму на текущем листе; на диаграмме должен быть отображен заголовок, подписи данных, легенда.

Тема: Электронные презентации PowerPoint.

Задания к лабораторной работе:

1. Вставить любым способом пять пустых слайдов.

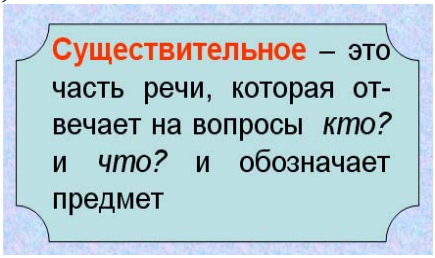
2. Применить ко всем слайдам способ заливки – текстура (любую понравившуюся).

3. Для первого слайда выбрать автомакет – титульный слайд. Применить к нему следующее оформление:

– Текст заголовка – РУССКИЙ ЯЗЫК, шрифт – Monotype Corsiva размер – 80, цвет – синий, начертание – жирный.

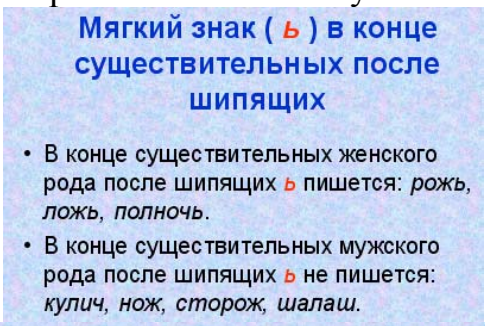
– Текст подзаголовка – ИМЯ СУЩЕСТВИТЕЛЬ-НОЕ, шрифт – Arial, размер – 0, цвет – красный, начертание – полужирный курсив, тень.

4. Для второго слайда выбрать автомакет – только заголовок и набрать приведенный текст (см. слайд). Применить к нему следующее оформление: шрифт – Arial. размер – 46, цвет: слово «существительное» – красный, остальной текст – черный;



начертание: слово «существительное» – жирный, остальной текст – обычный. Весь текст находится внутри рамки (см. слайд), цвет заливки рамки – светло голубой.

5. Для третьего слайда автомакет – схема и оформить его следующим образом (см. слайд). Цвет заливки элементов схемы – светло зеленый, цвет шрифта для «имя существительное» - темно синий, остальной текст - черный. Обратите внимание на начертание некоторых слов.



6. Для четвертого слайда выбрать автомакет – заголовок и маркированный текст и набрать приведенный текст (см. слайд). Цвет заголовка – синий, «ь» в заголовке и по тексту маркированного списка выделен красным цветом. Обратите внимание на начертание некоторых слов.

7. Для пятого слайда выбрать автомакет – заголовок и таблица и набрать приведенный текст (см. слайд). Цвет заголовка – красный. Обратите внимание на начертание некоторых слов.

Название падежа	Сокращенное обозначение	Вопросы падежа	Предлоги
Именительный	*Им.п.*	*кто? что?*	–
Родительный	*Р.п.*	*кого? чего?*	без, для, у, с, от, до, из
Дательный	*Д.п.*	*кому? чему?*	к, по
Винительный	*В.п.*	*кого? чего?*	в, на, за, под, через
Творительный	*Тв.п.*	*кем? чем?*	за, над, под, с, перед
Предложный	*Пр.п.*	*о ком? о чем?*	о, об, в, во, на

8. Вставить на первом слайде в верхний левый угол  следующую картинку (если таковой не окажется в списке картинок, то можно вставить любую другую).

9. Задать для созданных слайдов различную анимацию и смену слайдов.

10. Установить для слайдов следующие управляющие кнопки:

– первый слайд:  – далее и  – в конец;
– последний слайд:  – в начало;
– остальные слайды:  – далее.

10. Осуществить показ созданной презентации.

11. Сохранить презентацию с именем слайды.pps.

Тема: СУБД Access.

Тема №1: Создание таблиц базы данных.

Задания к лабораторной работе:

1. Загрузите *Access*, в открывшемся окне Microsoft Access щелкните селекторную кнопку *Новая база данных*, затем - кнопку OK.

2. В открывшемся окне Файл новой базы данных в поле Имя файла: введите имя *Кафедра*, в раскрывающемся списке поля Папка: выделите свою папку на диске C:, затем закройте окно, щелкнув кнопку Создать. Создаваемая база данных будет находиться в вашей папке на диска D:.

3. В основном окне базы данных Кафедра: база данных выберите вкладку *Таблицы* и щелкните кнопку Создать.

4. В открывшемся окне Новая таблица выберите пункт *Конструктор* и щелкните кнопку OK. В результате откроется окно Таблица1: таблица в режиме *Конструктор*, в котором определите поля таблицы согласно Таблице 3. Для этого:

- введите в строку столбца *Имя поля* имя *Код преподавателя*,
- поместите курсор в строку столбца *Тип данных*, раскройте список и выберите в нем слово *Счетчик*. В строку столбца *Описание* можно ничего не вводить,
- аналогично определите другие имена и тип данных полей таблицы,
- в нижней части таблицы *Свойства поля* во вкладке *Общие* установите указанные в Таблице 1 размеры полей, предварительно поместив курсор на соответствующую запись (строку) в верхней части таблицы.

Таблица 1. Структура таблицы данных *Преподаватели*

Имя поля	Тип данных	Размер поля
Код преподавателя	Счетчик	
Фамилия	Текстовый	15
Имя	Текстовый	15
Отчество	Текстовый	15
Должность	Текстовый	9
Дисциплина	Текстовый	11
Зарплата	Денежный	

5. Сохраните созданную таблицу под именем *Преподаватели*.

6. На запрос о задании ключевых полей ответьте *Нет*.

7. Вернемся к таблице *Преподаватели* в режиме *Конструктор*. Если ее нет на экране, то активизируйте вкладку *Таблицы* и откройте ее.

8. В верхней части таблицы щелкните по записи *Должность*.

9. В нижней части окна *Свойства поля* во вкладке *Общие* щелкните по строке параметра *Условие на значение* и установите ограничения на данные, которые будут вводиться в поле Должность: должны вводиться только слова *Профессор*, *Доцент* или *Ассистент*. Для этого:

- щелкните по кнопке , чтобы открыть окно Построитель выражений,
- в открывшемся окне введите слово *Профессор*, щелкните по кнопке  , добавьте *Доцент*, снова щелкните по кнопке  , затем добавьте *Ассистент* и за-

кройте окно, щелкнув кнопку ОК.

– в строке параметра *Сообщение об ошибке* введите сообщение: *Такой должности нет*,

– в строке параметра *Значение по умолчанию* введите *Доцент*.

10.Введите ограничения на данные в поле Код преподавателя. С одной стороны эти данные не должны повторяться, а с другой - должна быть обеспечена возможность их изменения. Поэтому в этом поле нельзя использовать тип данных *Счетчик*, т.к. значения таких полей обновлять нельзя. Для выполнения второго условия задайте тип данных *Числовой*. А для выполнения первого условия выполните следующее:

– щелкните по строке параметра *Индексированное поле*,

– раскройте список и выберите пункт *Да (совпадения не допускаются)*,

– перейдите в режим *Таблица*, щелкнув кнопку  слева на панели инструментов и при запросе на сохранение щелкните кнопку Да.

11.В режиме *Таблица* заполните таблицу данными о преподавателях в соответствии с Таблицей 2 и проверьте реакцию системы на ввод неправильных данных в поле Должность, введя, например, слово *Лаборант*.

Таблица 2. Таблица данных *Преподаватели*

Код	Фамилия	Имя	Отчество	Должность	Дисциплина	Зарплата
1	Куклев	Федор	Петрович	Доцент	Информатика	9800 р.
2	Максимов	Никита	Юрьевич	Профессор	Экономика	14500 р.
3	Андреев	Борис	Сергеевич	Доцент	Статистика	7600 р.
4	Лидский	Виктор	Борисович	Профессор	Математика	12500 р.
5	Колосов	Алексей	Ивановна	Доцент	Математика	8900 р.
6	Беляев	Виктор	Павлович	Ассистент	Информатика	3900 р.
7	Максимов	Иван	Николаевич	Доцент	Информатика	8900 р.

12. Измените ширину поля Дисциплина в соответствии с шириной данных, предварительно переведя курсор в любую строку данного поля и используя команду Формат/Ширина столбца... - кнопка По ширине данных.

13. Произведите поиск преподавателя Максимова, для чего поместите курсор в первую строку под именем поля Фамилия и выберите команду Правка/Найти....

14. В открывшемся диалоговом окне Поиск в поле: Фамилия в поле Образец: введите *Максимов*, в поле Просмотр: выберите *Все*, в поле Совпадения: выберите *С любой частью поля*, у опции Только в текущем поле установите флажок и щелкните кнопку Найти. Чтобы найти следующую запись, щелкните кнопку Найти далее. Затем щелкните кнопку Закрыть.

15. Измените заработную плату ассистенту Беляеву с 3900руб. на 4300руб., для чего поместите курсор в первую строку поля Зарплата и выберите команду

Правка/Заменить....

16. В открывшемся диалоговом окне Замена в поле: Зарплата в поле Образец: введите *3 900р.*, в поле Заменить на: введите *4 300р.*, в поле Просмотр: выберите *Все*, у опции Только в текущем поле установите флажок и щелкните кнопку Найти далее. Когда запись будет найдена, щелкните кнопку Заменить. Затем щелкните кнопку Закрыть.

17. Произведите сортировку данных в поле Зарплата по убыванию, используя команду Записи/Сортировка/Сортировка по убыванию (предварительно поместив курсор на любую запись данного поля).

18. Произведите фильтрацию данных сначала по полю Должность (например, профессор), затем – Дисциплина (например, математика), используя команду Записи/Фильтр/Фильтр по выделенному (предварительно поместив курсор на соответствующую запись данного поля). После выполнения окончательной фильтрации удалите фильтр командой Записи/Удалить фильтр.

19. Просмотрите, как созданная таблица будет выглядеть на листе бумаги при печати, используя команду Файл/Предварительный просмотр.

20. Закройте таблицу *Преподаватели*, щелкнув кнопку  .

21. Создайте в базе данных *Кафедра* таблицу *Студенты*. Для этого:

– в окне базы данных выберите вкладку *Таблицы* и щелкните по кнопке Создать;

– в открывшемся окне Новая таблица выберите пункт *Конструктор* и щелкните по кнопке ОК;

– создайте структуру таблицы *Студенты*: определите ее имена полей, тип данных и свойства полей в соответствии с Таблицей 3;

– в качестве ключевого поля определите поле Код студента, для чего активизируйте это поле и щелкните по кнопке  ;

– закройте таблицу, задав ей имя *Студенты*.

Таблица 3. Структура таблицы данных *Студенты*

Имя поля	Тип данных	Размер поля
Код студента	Числовой	Целое
Фамилия	Текстовый	15
Имя	Текстовый	15
Отчество	Текстовый	15
Номер группы	Числовой	Целое
Телефон	Текстовый	9
Стипендия	Логический	Да/Нет

22. Аналогично создайте структуру таблицы *Дисциплины*, определив ее имена полей, тип данных и свойства полей в соответствии с Таблицей 4, а ключевым выбрав поле Код дисциплины.

Таблица 4. Структура таблицы данных *Дисциплины*

Имя поля	Тип данных	Размер поля
Код дисциплины	Числовой	Целое
Название дисциплины	Текстовый	30

23. Создайте в базе данных *Кафедра* структуру таблицы *Оценки*, определив ее имена полей, тип данных и свойства полей в соответствии с Таблицей 5. На во-

прос о создании ключевого поля ответьте Нет.

Таблица 5. Структура таблицы *Оценки*

Имя поля	Тип данных	Размер поля
Код студента	Числовой	Целое
Код дисциплины	Числовой	Целое
Оценки	Числовой	Байт

24. В базе данных *Кафедра* измените структуру таблицы *Преподаватели*, добавив в нее поле Код дисциплины и определив его тип данных и размер в соответствии с данными Таблицей 4.

25. Закройте таблицы *Преподаватели*, *Дисциплины*, *Оценки* и *Студенты*, если они открыты.

26. Щелкните по кнопке  на панели инструментов (или выполните команду Сервис/Схема данных). На экране откроется окно Добавление таблицы.

27. В открывшемся окне будет выделено название одной таблицы. Щелкните по кнопке Добавить.

28. Переведите выделение на имя следующей таблицы и щелкните по кнопке Добавить. Аналогично добавьте оставшиеся две таблицы.

29. Закройте окно, щелкнув по кнопке  .

30. Создайте связь между таблицами *Дисциплины* и *Оценки*. Для этого подведите курсор мыши к полю Код дисциплины в таблице *Дисциплины*, щелкните левой кнопкой мыши и, не отпуская ее, перетащите курсор на поле Код дисциплины в таблицу *Оценки*, а затем отпустите кнопку мыши. На экране откроется окно Связи.

31. Установите флажок («галочку») в свойстве Обеспечение целостности данных, щелкнув по нему.

32. Установите флажок в свойстве Каскадное обновление связанных полей и Каскадное удаление связанных полей.

33. Щелкните по кнопке Создать. Связь будет создана.

34. Аналогично создайте связи между полями Код дисциплины в таблице *Дисциплины* и полем Код дисциплины в таблице *Преподаватели*, а также между полем Код студента в таблице *Студенты* и полем Код студента в таблице *Оценки*.

35. Закройте окно схемы данных, ответив Да на вопрос о сохранении макета.

Тема №2: Работа с формами.

Задания к лабораторной работе:

1. В окне базы данных Кафедра: база данных активизируйте вкладку *Формы* и щелкните кнопку Создать.

2. В открывшемся окне Новая форма выберите *Мастер форм*, а в нижней части окна раскройте список и выберите таблицу *Преподаватели*, затем нажмите кнопку OK.

3. В открывшемся окне Создание форм из поля списка Доступные поля переведите в поле списка Выбранные поля те, которые будут присутствовать в форме. В данном случае это все поля, поэтому щелкните по кнопке  и нажмите кнопку Далее.

4. В следующих окнах *Мастера форм* выберите соответственно внешний

вид формы *в один столбец*, затем требуемый стиль оформления и далее задайте имя форме *Состав преподавателей*.

5. После нажатия кнопки Готово откроется форма в один столбец.

6. Найдите запись о доценте Максимове, для чего сначала переведите курсор в первую строку поля *Фамилия*, затем выполните команду Правка/Найти.

7. Переведите курсор в первую строку поля Зарплата и, используя команду Правка/Заменить..., повысьте зарплату ассистенту с 4300руб. на 4500руб.

8. Произведите сортировку поля Фамилия по убыванию командой Запись/ Сортировка/Сортировка по убыванию, предварительно поместив курсор в любую запись данного поля.

9. Поместив курсор на запись *Доцент* поля Должность командой Запись/ Фильтр/Фильтр по выделенному, оставьте в таблице только записи о преподавателях - доцентах.

10. Аналогичной процедурой оставьте в таблице только записи о преподавателях – доцентах по дисциплине информатика.

11. Удалите фильтр командой Записи/Удалить фильтр.

12. Теперь измените название поля Дисциплина на Преподаваемая дисциплина. Для этого:

– перейдите в режим *Конструктор*, щелкнув кнопку  слева на панели инструментов,

– щелкните правой кнопкой мыши в поле Дисциплина (оно слева),

– в контекстном меню выберите пункт *Свойства*,

– в открывшемся окне свойств поля Дисциплина в строке *Подпись* вместо *Дисциплина* введите *Преподаваемая дисциплина*,

– для просмотра результата перейдите в режим Форма.

13. Просмотрите, как данная форма будет выглядеть на листе бумаги.

14. Закройте форму *Состав преподавателей*, щелкнув кнопку  .

Тема №3: Формирование запросов и отчетов для отдельных таблиц базы данных.

Задания к лабораторной работе:

1. На основе таблицы *Преподаватели* создайте простой запрос на выборку, в котором должны отображаться фамилии, имена, отчества преподавателей и их должность. Для этого:

– откройте вкладку *Запросы* и в открывшемся окне щелкните по кнопке Создать,

– в окне Новый запрос выберите пункт *Простой запрос* и нажмите кнопку ОК,

– в открывшемся окне Создание новых запросов в строке *Таблицы/запросы* выберите таблицу *Преподаватели*, а в поле списка *Доступные поля* выделите слово *Фамилии* и щелкните по кнопке  для перевода его в поле *Выбранные поля*,

– аналогично переведите туда поля *Имя*, *Отчество*, *Должность* и щелкните по кнопке Далее,

– в строке параметра *Задайте имя запроса* введите имя *Должности преподавателей* и щелкните по кнопке Готово.

2. В открывшейся таблице с результатами запроса отсортируйте по убыванию данные по полю Должность.

3. Сохраните запрос.

4. Создайте новый запрос под именем *Преподаваемые дисциплины* на выборку с параметром, в котором должны отображаться фамилии, имена, отчества преподавателей и преподаваемые ими дисциплины, а в качестве параметра задайте фамилию преподавателя и выполните этот запрос для преподавателя *Лидского*. Для этого:

– создав аналогично предыдущему пункту запрос под именем *Преподаваемые дисциплины*, перейдите в режим Конструктор,

– в строке параметра *Условия отбора* для поля Фамилия введите фразу: *[Введите фамилию преподавателя]*,

– выполните запрос командой Запрос/Запуск,

– в открывшемся окне введите фамилию *Лидский* и щелкните по кнопке ОК.

5. Сохраните запрос и закройте окно запроса.

Тема: Системы счисления.

1) Перевести двоичное число вовсе известные вам системы счисления:

а) 1001011101
б) 10110001111
в) 1111011010
г) 1111100001
д) 100011100011
е) 10001101001
ж) 111100000111111
з)10101100110101
и) 1111000111110101
к) 10101101011010101

2) Перевести восьмеричное число вовсе известные вам системы счисления:

а) 526
б) 457
в) 562
г) 125
д) 443
е) 361
ж) 777
з) 1267
и) 6375
к) 774527

3) Перевести десятичное число вовсе известные вам системы счисления:

а) 58
б) 96
в) 129
г) 345
д) 789
е) 953
ж) 1283
з) 1892
и) 5638
к) 105896

4) Перевести шестнадцатеричное число вовсе известные вам системы счисления:

а) 1A
б) 26
в) 3AF
г) C45
д) D56
е) AFD
ж) 4A5F
з) 9E6CA
и) ABC5F
к) 48FF56A

Тема: Основы программирования в С++

Тема №1 **Линейные программы**

Задание. Написать программу для расчета двух формул. Подготовить тесто-

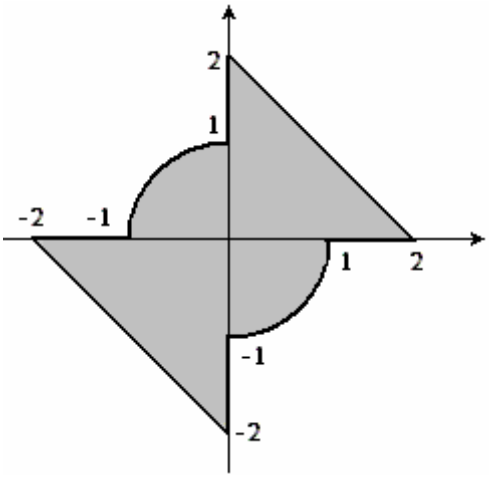
вые задания и проверить результат с помощью калькулятора. Результаты вычислений по первой и второй формуле должны совпадать.

Вариант 1 $z_1 = 2\sin^2(3\pi - 2\alpha)\cos^2(5\pi + 2\alpha)$ $z_2 = \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right)$	Вариант 4 $z_1 = \cos^4 x + \sin^2 y + \frac{1}{4}\sin^2 2x - 1$ $z_2 = \sin(x+y)\sin(y-x)$
Вариант 2 $z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha$ $z_2 = 2\sqrt{2}\cos\alpha\sin\left(\frac{\pi}{4} + 2\alpha\right)$	Вариант 5 $z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2$ $z_2 = -4\sin^2\frac{\alpha-\beta}{2}\cos(\alpha+\beta)$
Вариант 3 $z_1 = \frac{1 - 2\sin^2\alpha}{1 + \sin 2\alpha}$ $z_2 = \frac{1 - tg\alpha}{1 + tg\alpha}$	Вариант 6 $z_1 = \frac{\sin\left(\frac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$ $z_2 = ctg\left(\frac{5}{4}\pi + \frac{3}{2}\alpha\right)$

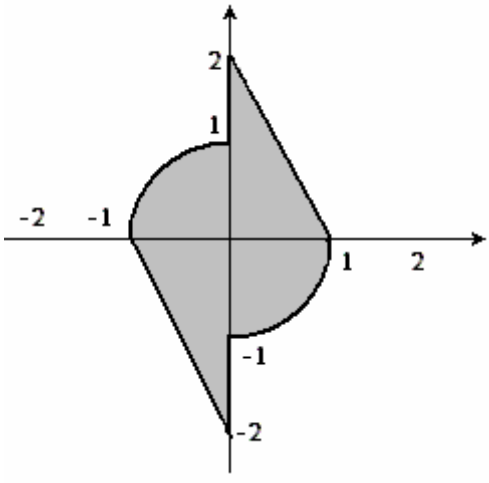
Тема № 2 Программирование разветвляющей структуры

Задание. Дана «мишень» в виде закрашенной области, изображенной на рисунке. Создать алгоритм для определения попадания точки с координатами (x, y) в мишень. Значение координат точки вводить с клавиатуры. Написать программу с использованием условного оператора if.

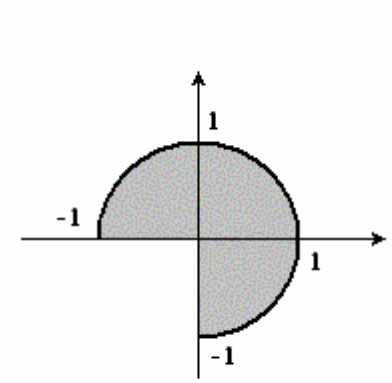
1.



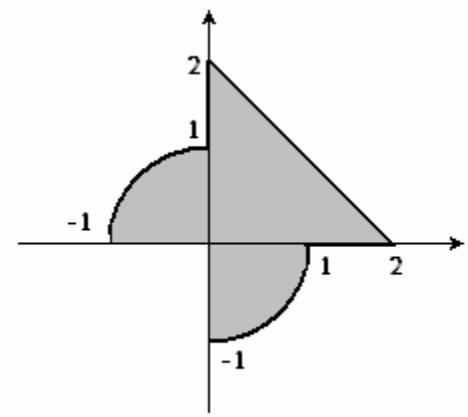
2.



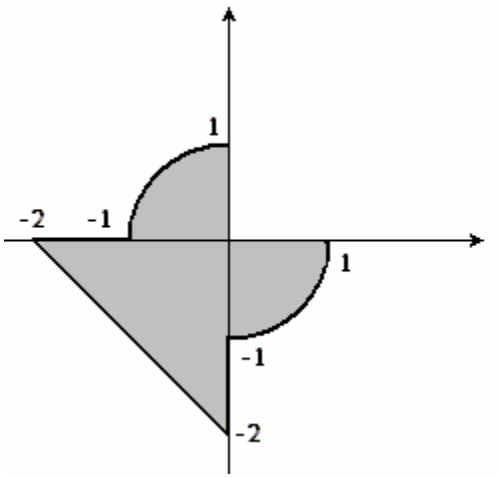
3.



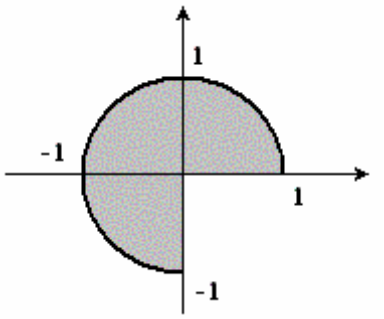
4.



5.



6.



Тема №3 **Разветвляющиеся программы. Оператор switch**

Задание. Написать программу с использованием оператора множественного выбора switch.

Варианты

1. Создать программу, вычисляющую площадь фигуры на плоскости.

$$S=\begin{cases} \pi R^2, & при \quad k=1 \\ a\sqrt{3}/4, & при \quad k=3 \\ a^2, & при \quad k=4 \\ 3\sqrt{3}a/2, & при \quad k=6 \end{cases}$$

Значение k вводить с клавиатуры. Предусмотреть вывод сообщения при вводе отсутствующего значения k.

2. С клавиатуры ввести число y, обозначающее день недели. В зависимости от введенного значения на экран вывести название дня недели. Предусмотреть вывод об ошибке ввода, и также напоминание о выходных днях.

3. С клавиатуры ввести число x, обозначающее номер месяца. В зависимости от введенного значения на экран вывести название сезона (лето, весна, осень, зима). Предусмотреть вывод об ошибке ввода, если введенного число не соответ-

ствуют возможному значению номера месяца.

4. Создать программу, содержащую справочную информацию о значениях тригонометрических функций. По введенной с клавиатуры величине угла x, выраженной в градусах, программа должна выдавать соответствующие значения cos x и sin x.

градусы	0	30	45	60	90
sin x	0	1/2	$\sqrt{2}/2$	$\sqrt{3}/2$	1
cos x	1	$\sqrt{3}/2$	$\sqrt{2}/2$	1/2	0

Предусмотреть вывод об ошибке ввода, если введенного число не соответствуют возможному значению.

5. С клавиатуры ввести число m, обозначающее номер месяца. В зависимости от введенного значения программа должна выводить на экран количество дней в месяце (без учета високосных года). Предусмотреть вывод об ошибке ввода, если введенного число не соответствуют возможному значению номера месяца.

6. С клавиатуры ввести число m, обозначающее номер месяца. В зависимости от значения введенного числа программа должна выводить на знак по гороскопу: (1 - козерог, 2 - водолей, 3 – рыбы, 4 – овен, 5 – телец, 6 – близнецы, 7 – рак, 8 – лев, 9 – дева, 10 – весы, 11- скорпион, 12 – стрелец). Предусмотреть вывод об ошибке ввода, если введенного число не соответствуют возможному значению номера месяца.

Тема № 4. Программирование разветвляющей и циклической структуры

Задание. Вычислить и вывести на экран в виде таблицы все значения функции y на заданном интервале с шагом Δt. В каждой строке таблицы должно содержаться значение аргумента и значение функции. Таблица должна содержать заголовок и шапку. Для выравнивания границ таблицы при выводе на экран использовать библиотеку iomanip.h. Значение констант задать в программе. Для задания цикла в программе использовать оператор while. Составить блок-схему алгоритма.

Вариант 1

$$y=\begin{cases} at^2\ln t, & 1\le t\le 2 \\ 1, & t<1 \\ e^{at}\cos bt, & t>2 \end{cases} \quad \text{a = -0.5,} \quad \text{b = 2,} \quad t\in[0;3], \quad \Delta t = 0.15$$

Вариант 2

$$y=\begin{cases} \pi t^2 - 7/t^2, & t<1.3 \\ at^3+7\sqrt{t}, & t=1 \\ \lg(t+7\sqrt{t}), & t>1.3 \end{cases} \quad \text{a=1.5,} \quad t\in[0.8;2], \quad \Delta t = 0.1$$

Вариант 3

$$y=\begin{cases} at^2+bt+c, & t<1.4 \\ a/t+\sqrt{t^2+1}, & t=1.4 \\ (a+bt)/\sqrt{t^2+1}, & t>1.4 \end{cases}$$ a=2.8, b=-0.2, c=4 $t\in[1;2]$, $\Delta t=0.05$

Вариант 4

$$y=\begin{cases} \pi t^2-7t^2, & t<1.4 \\ at^3+7\sqrt{t}, & t=1.4 \\ \ln(t+7\sqrt{|t+a|}, & t>1.4 \end{cases}$$ a=1.65, $t\in[0.7;2]$, $\Delta t=0.1$

Вариант 5

$$y=\begin{cases} 1.5\cos^2 t, & t<2 \\ 1.8at, & t=2 \\ (2-t)+3tgt, & t>2 \end{cases}$$ a=2.3, $t\in[0.2;2.8]$, $\Delta t=0.2$

Вариант 6

$$y=\begin{cases} t\sqrt{t-a}, & t>a \\ t\sin at, & t=a \\ e^{-at}\cos at & t<a \end{cases}$$ a=2.5, $t\in[1;5]$, $\Delta t=0.5$

Тема №5 **Одномерные массивы**

Задание. Составить блок-схему алгоритма решения задачи. Написать программу на языке С++. Элементы массива ввести с клавиатуры.

Варианты

1. В одномерном массиве А, содержащем 15 элементов, определить сумму положительных элементов, расположенных после минимального элемента данного массива.
2. В одномерном массиве В, содержащем 12 элементов, определить среднее арифметическое отрицательных элементов, расположенных после минимального элемента данного массива.
3. В одномерном массиве С, содержащем 14 элементов, определить произведение положительных элементов, расположенных до максимального элемента этого массива.
4. В одномерном массиве А, содержащем 14 элементов, определить сумму положительных элементов, расположенных между минимальным и максимальным элементами данного массива.
5. В одномерном массиве В, содержащем 16 элементов, определить среднее геометрическое отрицательных элементов, расположенных между минимальным и максимальным элементами данного массива.
6. В одномерном массиве С, содержащем 13 элементов, определить сумму минимального и максимального элементов данного массива.

Тема № 6 **Символьные массивы**

Задание. Символьная строка является массивом типа int. Для работы с символьными строками воспользуйтесь функциями библиотеки string.h

Варианты

1. Одну строку инициализировать в программе, другую – ввести с клавиатуры. Сравнить данные строки по длине. Если они не равны, присоединить к меньшей строке - большую. Определить количество слов в полученной строке. Результат вывести на экран.

2. Ввести две строки с клавиатуры. Посчитать в каждой из них количество гласных букв. В строке, содержащей большее число гласных, удалить все согласные буквы.

3. Одну строку инициализировать в программе, другую – ввести с клавиатуры. Соединить их содержимое. Определить длину полученной строки. Вывести на экран первую половину полученной строки.

4. Одну строку инициализировать в программе, другую – ввести с клавиатуры. Если их содержимое одинаково, оставить строки без изменения, иначе соединить содержимое строк. При этом первыми должны быть символы той строки, в которой первый символ по алфавиту раньше.

5. Одну строку инициализировать в программе, другую – ввести с клавиатуры. В каждой из строк исключить поместить первый символ в конец строки, после чего соединить строки, расположив вначале большую по алфавиту.

6. В строке символов найти самое длинное и самое короткое слово. Сформировать новую строку, расположив в ее начале самое короткое слово, затем самое длинное, а потом все остальные слова в алфавитном порядке.

Тема №7 ***Двумерные массивы***

Задание. Составить блок-схему алгоритма решения задачи. Написать программу на языке С++. Значения элементов массива вводить с клавиатуры. Осуществить вывод элементов первоначальной и измененной матрицы в общепринятом виде (в виде таблицы).

Варианты

1. В матрице размерности 8 на 6 определить номер первого из столбцов, содержащих хотя бы один нулевой элемент. В каждой второй строке матрицы заменить максимальный элемент нулем. Найти сумму положительных элементов матрицы.

2. В матрице размерности 5 на 10 определить количество столбцов, содержащих хотя бы один нулевой элемент, вывести на экран их номера. В каждой строке матрицы, поменять местами максимальный и минимальный элементы. Заменить положительные элементы последний строки нулями.

3. В матрице размерности 8 на 8 определить сумму элементов в столбцах, не содержащих отрицательные элементы. Найти минимум среди сумм модулей элементов диагоналей, параллельных главной диагонали матрицы.

4. В матрице размером 8 на 8 найти такие k, что k-ая строка матрицы совпадает с k-ым столбцом. Увеличить минимальный элемент матрицы в десять раз.

Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.

5. В матрице размером 6 на 9 определить сумму модулей его отрицательных нечетных элементов каждой строки. Найти сумму элементов в тех столбцах, которые содержат хотя бы один нулевой элемент. Заменить положительные элементы второго и третьего единицами.

6. В матрице размером 10 на 10 найти сумму элементов в тех столбцах, которые содержат хотя бы один нулевой элемент. Заменить отрицательные элементы матрицы их модулями. В измененной матрице найти произведение элементов, расположенных ниже главной диагонали.

Тема №7 **Функции**

Задание. Составить пользовательскую функцию. Продемонстрировать ее использование в программе.

Варианты

1. Определить среднее арифметическое сторон двух треугольников, заданных координатами их вершин. Длину стороны треугольника вычислять в функции.

2. Вычислить среднее арифметическое объемов шаров с радиусами r_1, r_2, r_3. Для нахождения объема шара использовать функцию.

3. Заданы три конуса (радиусы основания и высота). Определить конус с наибольшим объемом. Для нахождения объема конуса использовать функцию.

4. Вычислить значение $z = (sign\, x + sign\, y)\, sign\,(x + y)$,

где $sign\, a = \begin{cases} -1, & при \quad a < 0, \\ 0, & при \quad a = 0, \\ 1, & при \quad a > 0. \end{cases}$

5. Составить программу вычисления значений:

$$a = \frac{\sqrt{z^3 + 4z^2 + 7z + 1}}{2 + e^{2z^3 + z - 3}}, \quad b = \frac{t^2 + 13t + 16}{7t^3 + t^2 - 4}.$$

Для определения значений многочленов использовать функцию. Значения величин z, t вводить с клавиатуры.

6. Вычислить значение $f = \dfrac{\max(a,b,c) * \min(a,c,d) - \max(b,c,d)}{\min(a,b,c)}$,

где a, b, c, d – некоторые значения, введенные с клавиатуры. Для нахождения максимального и минимального значений использовать функции.

Тема №8 **Массивы и указатели**

Задание. В каждом задании требуется создать функцию, параметром которой является одномерный массив. Продемонстрируйте вызов функции. Тело функции задайте двумя способами: обращение к элементам через указатели и обычным способом. Продемонстрируйте возможность применения созданной функции к строкам двумерного массива.

Варианты

1. Создайте функцию, меняющую местами минимальный и максимальный элементы одномерного массива. Воспользуйтесь функцией для каждой строки матрицы B (5, 4).

2. Создайте функцию, вычисляющую среднее арифметическое отрицательных элементов одномерного массива. Воспользуйтесь данной функцией для каждой строки матрицы A (4, 8).

3. Создайте функцию, вычисляющую сумму отрицательных элементов одномерного массива в случае, если их более 3, и возвращающую нуль иначе. Воспользуйтесь данной функцией для каждой строки матрицы X (4, 7).

4. Создайте функцию, вычисляющую сумму значений элементов одномерного массива, расположенных между его максимальным и минимальным элементами. Воспользуйтесь функцией для каждой строки матрицы O (3, 8).

5. Создайте функцию, подсчитывающую количество отрицательных элементов одномерного массива, порядковый номер которых меньше номера минимального элемента. Воспользуйтесь функцией для каждой строки с четным номером матрицы A (8, 4).

6. Создайте функцию, заменяющую положительные элементы массива нулями, а отрицательные их абсолютными величинами, увеличенными в 10 раз. Воспользуйтесь функцией для каждой нечетной строки матрицы C (5, 6).

Тема №9 **Ссылки** (2 часа)

Задание. В алгоритме решения задачи выделите необходимые функции (как правило, повторяющаяся последовательность действий). Функция должна возвращать несколько значений, для этого требуется использование ссылок. В программе продемонстрируйте вызов функции.

Варианты

1. Расположите в порядке возрастания корни квадратных уравнений $x^2 + bx - 4 = 0$ и $ax^2 - 8 + c = 0$ (где ***a, b, c*** – значения, вводимые с клавиатуры). В случае мнимых корней считать их равными нулю.

2. Даны два одномерных массива вещественных чисел C(12) и B(10). Вычислить значение $z = \frac{S_A^+ * S_B^-}{S_B^+ + S_A^-}$, где S_A^+ – сумма положительных элементов вектора A. Сумму положительных и отрицательных элементов для одного вектора находить в одной функции.

3. Даны массивы K(14) и C(8). Вычислить значение $z = \frac{S_A^+ * S_C^+}{K_C^+ + K_A^+}$, где S_A^+, K_A^+ – сумма и количество положительных элементов вектора A. Сумму и количество положительных элементов для одного вектора находить в одной функции.

4. Создайте функцию, подсчитывающую по заданным сторонам треугольника величины его углов. Выведите значения полученных углов из главной функ-

ции.

5. Создайте функцию, получающую в качестве аргументов некоторый вес, выраженный в килограммах и граммах, а также величину увеличения данного веса в граммах. Функция должна возвращать новый вес, выраженный в граммах и килограммах.

6. Создайте функцию, получающую в качестве аргументов длины сторон треугольника и подсчитывающую его периметр и площадь. Продемонстрируйте работу данной функции на примере трех треугольников, и расположите величины их площади по возрастанию.

VIII. КАРТА ОБЕСПЕЧЕННОСТИ ДИСЦИПЛИНЫ КАДРАМИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА

Ф.И.О. должность	Ученая степень и ученое звание	Вид занятия
Галаган Т.А., доцент	к.т.н., доцент	Лекция
Барабаш С.А., ассистент		Лабораторная работа

СОДЕРЖАНИЕ

Татьяна Алексеевна Галаган,
доцент кафедры ИиУС АмГУ
Татьяна Анатольевна Макарчук,
доцент кафедры ОМиИ АмГУ

Информатика. Учебно-методический комплекс для спец. 220302 – «Автоматизация технологических процессов и производств»

Изд-во АмГУ. Подписано к печати Формат 60x84/16. Усл. печ. л.
Тираж Заказ