МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

КЕМЕРОВСКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ ПИЩЕВОЙ ПРОМЫШЛЕННОСТИ

А.Г.СЕМЕНОВ

ИНФОРМАТИКА

1. Общая информатика.

Учебное пособие для студентов всех специальностей
всех форм обучения

КЕМЕРОВО 2008

УДК 681.3 (075.8)

Информатика / А.Г.Семенов. – Кемерово: КемТИПП, 2006. – ч.1. Общие вопросы информатики. - 96 с.

ISBN 5-89289-125-9

Учебное пособие включает описание операционной системы ОС Windows и языка программирования Паскаль (включая основы алгоритмизации обработки данных). Изложение материала ориентировано на практическую работу студентов. Пособие снабжено иллюстрациями и примерами.

Ил. – 24. Табл – 12. Библ. назв. – 2.

Печатается по решению Редакционно-издательского совета Кемеровского технологического института пищевой промышленности.

Рецензенты:

- зав. кафедрой автоматизации исследований и технической кибернетики Кемеровского государственного университета, д.т.н., проф. В.Я.Карташов;
- к. ф.-м. н., доцент кафедры вычислительной техники и информационных технологий Кемеровского института Московского государственного университета коммерции В.С.Черкасов.

$$С\frac{2404090000}{У50(03)-02}$$

ISBN 5-89289-125-9

СОДЕРЖАНИЕ

I. РАБОЧАЯ ПРОГРАММА КУРСА "ИНФОРМАТИКА"

1.1. Цель изучения дисциплины.

Целью изучения курса информатики является получение знаний по следующим разделам: понятие информации, общая характеристика процессов сбора, передачи, обработки и накопления информации; технические средства реализации информационных процессов, алгоритмизация и программирование, языки программирования высокого уровня; операционные системы; пакеты прикладных программ общего назначения; текстовые и графические редакторы; табличные процессоры; базы данных; пакеты программ для организации работ в офисе; локальные вычислительные сети, компьютерная сеть Internet; справочные информационные системы; защита информации.

1.2. Содержание дисциплины.

1.2.1. Теоретическая часть.

	Тема	Часов	Литература
	Понятие информации, общая характеристика процессов сбора, передачи, обработки и накопления информации. История развития компьютеров. Технические средства реализации информационных процессов. Структура ЭВМ. Файловая система хранения и передачи информации, понятие папки, дерево папок.	Число часов зависит от учебного плана конкретного потока	[1], с.11-85
	Программное обеспечение персональных компьютеров. Пакеты прикладных программ. Операционные системы. Принципы организации ОС Windows. Работа с файлами в ОС Windows.	-*-	[1], с. 98-130, 166-185, 217-221, 364-382

	Архивирование файлов. Компьютерные вирусы. Антивирусные программы.		
	Алгоритмизация и программирование. Понятие алгоритма. Виды его представления. Языки программирования высокого уровня. Основные понятия языка Pascal.	-*-	[1], 560 - 607, [2], с.12 – 20
	Простые операторы языка Pascal. Условные операторы. Циклы. Операторы организации циклов. Работа с массивами.		[2], с. 20 – 50, 62 - 80
	Компьютерные сети. Internet. Защита информации.	-*-	[1], с. 198-225, 228-238

1.2.2. Тематика лабораторных работ.

1) Программирование линейного вычислительного процесса;
2) Циклический процесс. Обработка одномерного массива;
3) Обработка двумерного массива.
4-5)Работа с файлами с помощью ОС Windows.

1.3. Организация изучения дисциплины

Согласно учебному плану, изучение информатики происходит в течение одного семестра (двух сессий). В течение первой сессии прослушивается теоретический курс и выполняется часть лабораторных работ. В промежутке между сессиями студенты должны самостоятельно изучить материал, опираясь на прослушанные лекции, и выполнить контрольные работы. На второй сессии выполняется остаток лабораторных работ и происходит сдача экзамена.

Если учебный план предусматривает выполнение одной контрольной работы, в нее включаются задания 1 из контрольной работы № 1 и задание 2 из работы № 2.

1.4. Рекомендуемая литература

В принципе, при изучении дисциплины можно использовать большинство учебников по информатике для ВУЗов, самоучителей по работе на компьютере и учебных курсов по программированию на языке Паскаль. Наиболее предпочтительными являются:

1. Информатика. Базовый курс / Симонович С.В. и др. – СПб: Питер, 2001. – 640 с.

2. Немнюгин С.А. TURBOPASCAL. Учебник. – СПб: Питер, 2001. – 496 с.

II. ЛЕКЦИОННЫЙ КУРС

2.1. ОБЩИЕ ПОНЯТИЯ ИНФОРМАТИКИ

2.1.1. Информация и ее цифровая запись

Информация – это сведения об окружающем мире, которые человек может получать, передавать, перерабатывать, хранить и использовать в своей деятельности. По мере развития человечества совершенствовались и множились способы обработки информации. Вначале для этого использовались только возможности отдельного человека – органы чувств для получения, речь для передачи и память для хранения информации. Затем для хранения и передачи информации стала применяться письменность. В XIX веке появились способы передачи информации на большие расстояния на основе оптических и электрических явлений – оптический, затем электрический телеграф, радио.

Попытки автоматизировать обработку числовой информации – вычисления – предпринимались с давних времен. Счеты были изобретены еще в античную эпоху. В XVII веке француз Блез Паскаль первым высказал идею создания вычислительной машины. В XIX в. были предприняты попытки создания работающей вычислительной машины, сформулированы основные принципы программирования. В XX веке, в период между двумя мировыми войнами, появились первые автоматические вычислительные устройства на базе электромеханических узлов (реле).

Первая электронная вычислительная машина была создана в США во время второй мировой войны. Она называлась ЭНИАК (ENIAC – Electronic Numerical Integrator And Calculator , т.е. "электронный числовой интегратор и вычислитель"), занимала много места и обладала весьма скромными возможностями. Однако принципы, заложенные в ее основу, были применены при конструировании более совершенных машин. Развитие вычислительной техники шло в следующих направлениях:

1. уменьшение габаритов,
2. расширение ресурсов (быстродействие, объем памяти и др.),
3. расширение возможностей обработки различных типов информации,

4. упрощение взаимодействия машины и человека.

Рассмотрим более подробно третий пункт. Конструкция и работа компьютера базируется на электронных устройствах, которые могут находиться в одном из двух устойчивых состояний (течет ток – не течет ток, заряжен конденсатор – не заряжен, и т.д.). Этим состояниям можно поставить в соответствие цифры 0 и 1. Поэтому в основе работы компьютера лежит т.н. *двоичная* система счисления, использующая только эти две цифры.

Запись числа в привычной нам десятичной системе ведется по принципу: каждая цифра показывает количество определенных степеней числа 10; показатель степени на единицу меньше, чем положение данной цифры в записи числа, считая справа налево. Например, число 23578:

$$23578 = 8 \cdot 10^0 + 7 \cdot 10^1 + 5 \cdot 10^2 + 3 \cdot 10^3 + 2 \cdot 10^4.$$

Аналогично расшифровывается запись числа в двоичной системе, только теперь каждая цифра числа показывает количество определенных степеней двойки:

$2^0 = 1$	$2^5 = 32$	$2^{10} = 1024$
$2^1 = 2$	$2^6 = 64$	$2^{11} = 2048$
$2^2 = 4$	$2^7 = 128$	$2^{12} = 4096$
$2^3 = 8$	$2^8 = 256$	$2^{13} = 8192$
$2^4 = 16$	$2^9 = 512$	$2^{14} = 1638$

- и так далее. Например, запись числа 1011 означает:

$$1011 = 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 = 1 + 2 + 8 = 11.$$

Таким способом можно записать любое *целое* число.

Вещественные числа (имеющие или могущие иметь дробную часть) для обработки компьютером сначала записываются в единообразной форме – с плавающей точкой

$$325.867 = 0.325867 \cdot 10^3.$$

Такое число хранится в памяти компьютера в виде двух целых – записанной в виде целого числа последовательности значащих цифр (в данном случае 325867) и показателя степени десятки.

Символьная информация (буквы, знаки препинания, математические знаки и др.) переводится в цифровую форму приданием каждому символу определенного номера. За основу берется созданная в США таблица символов ASCII (American Standard Code for Information Interchange – "Стандартный американский код для информационного обмена"). Она включает 128 символов. Еще 128 символов добавляют в разных странах для кодирования национальных алфавитов и других нестандартных знаков.

Для перевода в цифровую форму *графической* информации (рисунки, фотографии, чертежи и т.п.), изображение разбивается на отдельные точки (*пикселы*), расположенные достаточно близко друг к другу. Такие отдельные точки легко можно видеть, например, под лупой на газетных фотографиях. Далее в памяти компьютера запоминаются координаты каждого пиксела, его цвет (по номеру в особой таблице цветов), яркость (оцененная по условной шкале количеством баллов). Качество обработки графической информации определяется частотой расположения пикселов (она измеряется количеством пикселов, помещающихся на участке в 1 дюйм, т.е., 2.54 см), количеством различаемых компьютером цветов и т.д.

Современные компьютеры могут обрабатывать в цифровой форме и звуковую информацию (музыку, человеческую речь).

Поскольку любую информацию компьютер переводит в цифровую форму, обрабатывает и хранит в виде комбинаций двоичных цифр (нулей и единиц), можно измерить количество информации количеством необходимых для ее записи двоичных цифр. За единицу количества информации принят один *бит* – количество информации, для записи которой требуется одна цифра (0 или 1). Более употребительной является другая единица – 1 байт. Это 8 бит, т.е., информация, для записи которой требуется 8 двоичных цифр. Легко доказать, что с помощью 8 цифр можно сформировать $2^8 = 256$ различных комбинаций от 00000000 до 11111111. Это значит, например, что 1 байта достаточно, чтобы закодировать любой символ (таблицы ASCII и ее дополнения) или любое целое число от 0 до 255.

Для измерения больших объемов информации применяют кратные единицы:

1 килобайт (кБ) = 2^{10} байт = 1024 байт

1 мегабайт (МБ) = 1024 кБ

1 гигабайт (ГБ) = 1024 МБ.

2.1.2. Общая структура компьютера

Минимальная структура компьютера на сегодняшний день:

Центральным элементом является *системный блок.* В нем находятся

1. *Материнская плата* - панель, на которой размещены *процессор* – вычислительное и управляющее устройство, микросхемы оперативной памяти и некоторые другие устройства;
2. Блок питания;
3. *Винчестер* – жесткий магнитный диск большой емкости (десятки ГБ) для длительного хранения информации;
4. Дисководы (один или больше) для чтения и записи информации на гибкие магнитные диски (дискеты). В основном применяются дискеты размером 3.5 дюйма, емкостью 1.44 МБ;
5. Устройства для чтения информации с лазерных оптических дисков емкостью 600 – 700 МБ (CD-ROM; в последнее время появились устройства не только для чтения, но и для записи информации на такие диски – CD-RW);
6. Разъемы для подключения внешних устройств и др.

Для ввода информации в компьютер служат *клавиатура* и *мышь*. Клавиатура – это панель с набором клавиш. Нажатие клавиши приводит к замыканию комбинации контактов и созданию набора сигналов, интерпретируемых, как некий двоичный код (символа или какой-то команды). Мышь – электромеханический манипулятор в виде коробочки с двумя или тремя кнопками на верхней поверхности. Перемещение мыши по поверхности стола вызывает перемещение по экрану монитора метки-указателя, который можно навести на изображение какого-то объекта на экране и нажатием одной из кнопок вызвать определенные действия компьютера.

Для вывода информации служит *монитор* – устройство, похожее на телевизор, на экран которого выводится символьная или графическая информация.

Существуют дополнительные устройства для ввода информации, преимущественно графической, например, *сканеры*. Для вывода информации могут применяться *принтеры* (печатающие устройства), *плоттеры* (графопостроители) и др. Эти и другие внешние устройства подключаются к компьютеру через особые разъемы системного блока.

2.1.3. Файлы.

Информация хранится на носителях (винчестере, дискетах, лазерном диске и т.п.), будучи разбитой на отдельные порции, называемые *файлами*. Файл записывается на носитель или читается с него как единое неделимое целое.

Отдельные файлы можно объединить в группу, называемую *каталогом*, *папкой* или *директорией*. Каталог получает собственное имя. Несколько каталогов также можно объединить в каталог более высокого уровня (*внешний* или *родительский*), по отношению к которому составляющие его каталоги являются *вложенными* или *дочерними*.

Самый общий каталог, объединяющий всю информацию, помещенную на тот или иной носитель, называется *корневым*. Корневые каталоги обозначаются буквами, за которыми ставится двоеточие, например:

A: - корневой каталог дискеты, вставленной в данный момент в приемник дисковода;

C: - корневой каталог винчестера;

D: – корневой каталог лазерного диска, вставленного в приемник читающего устройства.

В зависимости от конфигурации компьютера, могут иметься и другие обозначения корневых каталогов. Фактически, эти обозначения относятся не к самим носителям, а к читающим устройствам. Если вставить в дисковод новую дискету, ее корневой каталог также получит обозначение **A:** .

Файл имеет следующие атрибуты (характеристики):

1. *Имя*. В современных компьютерах, работающих под управлением операционной системы Windows, имена могут иметь произ-

вольную длину и структуру, содержать пробелы или дефисы, набираться кириллицей. В старых компьютерах, управлявшихся операционной системой MS DOS, имена должны были обязательно начинаться с буквы и могли содержать не более 8 символов, в число которых входили только латинские буквы, цифры и знак подчеркивания: _;

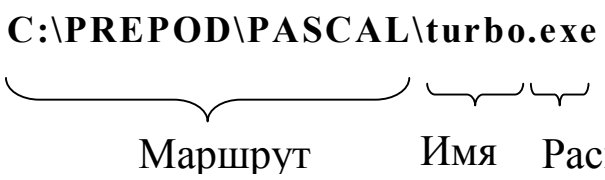
2. *Расширение* – комбинация из 2-3 букв, условно характеризующих содержимое файла, например:

txt, doc - файлы с текстовой информацией;
pas - программа, написанная на языке Паскаль;
exe - программа, записанная на "внутреннем" машинном языке (в двоичных кодах)

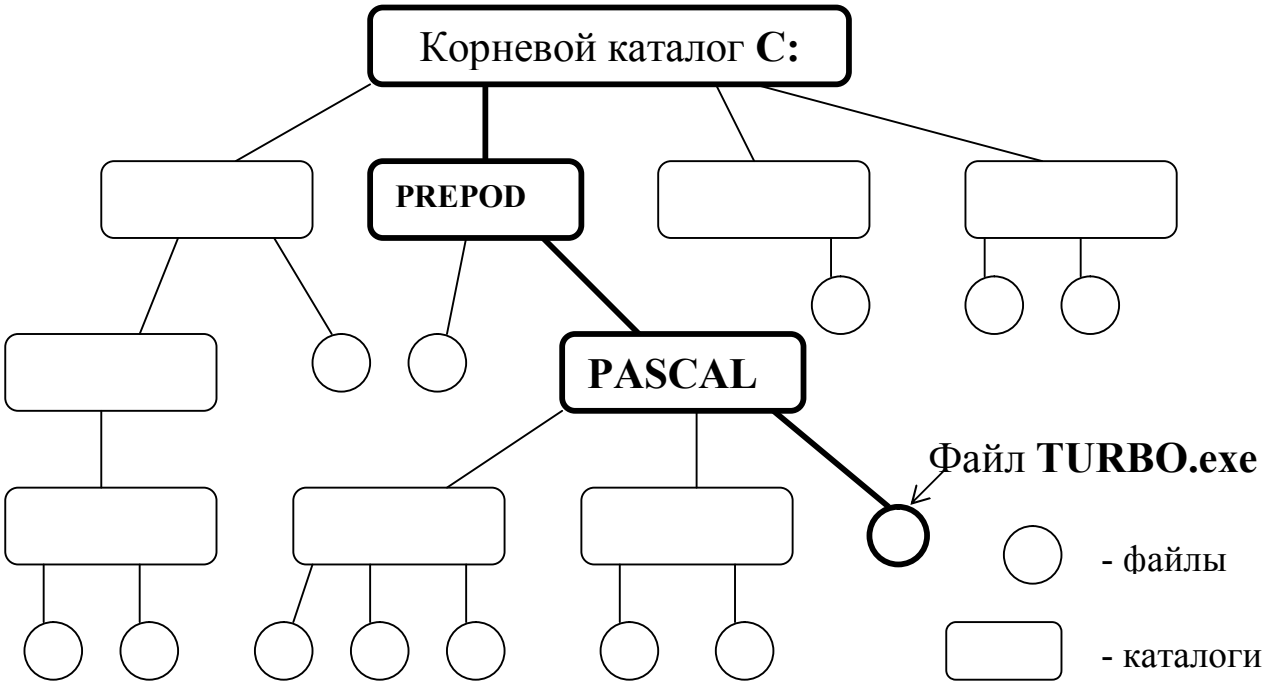
и так далее. Расширение ставится за именем файла и отделяется от него точкой.

3. *Путь или маршрут* – описание местонахождения файла в корневом каталоге в виде цепочки имен вложенных друг в друга каталогов, начиная с корневого и кончая каталогом, непосредственно содержащим данный файл. Отдельные имена каталогов разделяются косой чертой с обратным наклоном (она называется "*бэкслэш*"): \ .

Пример описания файла:



Общая структура файловой системы имеет вид дерева (дерево каталогов):



Выделен маршрут файла, описание которого приведено выше.

2.1.4. Виды программного обеспечения компьютеров.

Многообразные компьютерные программы можно разделить на несколько основных групп:

1. *Операционные системы (ОС)* – комплексы программ, обеспечивающих управление аппаратными средствами компьютера, исполнение программ других типов и диалог пользователя с компьютером. Совокупность средств, обеспечивающих диалог, называется *пользовательским интерфейсом ОС*.

2. *Системы программирования* – комплексы программных средств, обеспечивающие разработку и исполнение программ на каком-либо *языке программирования.* В состав современных систем программирования обычно входят:

- *Текстовый редактор,* обеспечивающий запись и редактирование программы на исходном языке;
- *Диспетчер файлов,* обеспечивающий поиск и чтение файлов с готовыми текстами программ, сохранение в виде новых файлов текстов написанных программ, уничтожение файлов и другие операции;

- *Компилятор* для перевода текста программы с исходного языка в машинный код с одновременной проверкой грамматических ошибок;
- *Запускатель* для выполнения программ и информирования об ошибках исполнения;

- и другие компоненты. Для одного и того же языка программирования могут существовать несколько конкурирующих систем программирования, различающихся возможностями, удобством внутреннего интерфейса и т.д.

3. *Прикладные пакеты* – программные комплексы, предназначенные для решения тех или иных специальных задач:

- *Текстовые процессоры* (Microsoft Word, Лексикон) предназначены для создания готовых текстовых документов различного содержания на компьютере. Они обеспечивают набор и редактирование текста, оформление абзацев, заголовков и других элементов текста, создание или включение в текст рисунков, таблиц и т.п. Более мощными являются *издательские системы* (Ventura Publisher, PageMaker и др.), предназначенные для компьютерного набора и верстки печатных изданий. Они предоставляют пользователю больший набор возможностей, включая автоматическое оформление элементов текста в различном стиле, создание оглавлений, предметных указателей и т.д. Специализированной издательской системой является система TeX для верстки книг и статей, насыщенных математической символикой;
- *Графические редакторы* для создания разнообразных графических документов. Их можно разбить на две группы:

1) *Векторные*, в которых электронный луч рисует на экране различные знаки – элементы изображения. Встроенными векторными редакторами снабжаются многие программные комплексы (редактор Paint в ОС Windows, графический редактор текстового процессора Word, модуль Graph в системе программирования TurboPascal, графические редакторы математических пакетов и др). Мощным графическим редактором общего назначения, основанным на векторном принципе, является Corel Draw. Векторными являются многие специализированные графические редакторы, такие, как системы инженерной графики AutoCad, Pro Engineer, система строительного черчения ArchiCad, система деловой графики Visio.
2) *Растровые,* в которых изображение непосредственно формируется из отдельных пикселов. Такой способ рисования обладает более

богатыми возможностями и реализуется в специализированных графических редакторах, таких как Adobe Photoshop.

- Системы управления базами данных (СУБД) – программы создания и обработки баз данных – структурированных наборов сведений о более или менее значительном количестве однотипных объектов. База данных может представлять собой список сотрудников предприятия, каталог книг библиотеки, или типов оборудования и т.п.

  Базы данных делятся на:

  – *табличные (реляционные),* в которых данные о каждом объекте составляют одну из строк (записей) некоторой таблицы. Внутри строки отдельные сведения (поля записи) как бы "равноправны" и однозначно связаны с данным объектом базы;

  – *иерархические*, в которых данные об объекте организуются в многоуровневую иерархическую структуру типа "дерева";

  – *сетевые*, для которых характерна связь объектов и данных по типу "многие ко многим".

Наиболее распространенными являются реляционные базы данных. Среди СУБД, основанных на реляционном принципе, наибольшее распространение получили система dBase и ведущие свое происхождение от нее FoxPRO, Paradox, Clipper; в последнее время популярной стала СУБД Access.

- *Табличные процессоры (электронные таблицы)* – программы обработки данных, организованных в виде таблиц. Такая структура данных предоставляет большие возможности их обработки, поэтому электронные таблицы являются весьма популярным инструментом решения задач различного профиля – технических расчетов, ведения всевозможных реестров (списков), обработки экономической и статистической информации, организации (без применения специальных СУБД) баз данных и др.
- *Математические пакеты* для проведения различных математических расчетов, решения задач математического моделирования физических и технических процессов. Существует большое количество различных математических пакетов: MathCad, MatLab, Derive, Maple и другие. Специализированный пакет Statistica предназначен для решения статистических задач, в том числе обработки экспериментальных данных.
- *Пакеты бухгалтерских расчетов* 1С:Бухгалтерия, БЭСТ и др.

Отдельные прикладные программы могут объединяться в *интегрированные пакеты*, для которых характерна взаимосвязь отдельных программ, возможность их совместной работы и обмена данными между ними. Примером такого пакета является Microsoft Office, объединяющий текстовый редактор Word, электронные таблицы Excel, СУБД Access и некоторые другие программы. Это дает возможность обработать таблицу Excel средствами СУБД, вставить таблицу в текст, созданный с помощью Word и т.п.

2.2. ОПЕРАЦИОННАЯ СИСТЕМА WINDOWS

2.2.1. Общие принципы организации ОС Windows

Для функционирования системы Windows требуется IBM-совместимый компьютер с процессором 80486 и выше, с оперативной памятью объемом 8 Мбайт и более, 60 Мбайт свободной памяти на жестком диске для полной установки системы (для минимальной установки — 30 Мбайт), дисплей SVGA, , манипулятор "мышь". Впрочем, эти данные относятся к ранним версиям ОС Windows, более современные версии ОС применимы только для более мощных машин.

Основные концепции ОС Windows:

1. Графический интерфейс с широким применением мыши.
2. Возможность одновременной работы с несколькими программами, действующими под управлением Windows (они называются приложениями Windows).
3. Независимость программ от внешних устройств.
4. Доступность всей оперативной памяти.
5. Динамическое подключение библиотек.
6. Связь и внедрение объектов(Object Linking and Embedding, OLE) - новый способ обмена данными между приложениями
7. Использование масштабируемых шрифтов True Type.

Основные средства управления ОС Windows:

1. *Меню* – готовый набор команд, предлагаемых компьютером для исполнения. Нужная команда меню выбирается с помощью мыши.

2. *Окно* – обрамленная часть экрана, служащая для уточнения параметров выполняемой команды.

3. *Панель инструментов* – набор *виртуальных* (то есть нарисованных на экране и "нажимаемых" с помощью мыши) кнопок, дублирующих отдельные команды меню.

4. *"Горячие" клавиши* – комбинация клавиш, нажимаемых одновременно (сначала нажимают первую, затем, не отпуская – вторую и если надо, третью). Нажатие такой комбинации вызывает выполнение какой-либо команды.

Таким образом, многие действия в ОС Windows можно выполнить разными способами – при помощи меню, панели инструментов и нажатием "горячих" клавиш. Особенности этих способов:

1. Использование "горячих" клавиш – самый быстрый способ выполнить команду, но запомнить большое количество разных комбинаций сложно. Горячими клавишами пользуются для выполнения наиболее часто применяемых команд или при ненадежной работе мыши.

2. Выполнение команд с помощью кнопок панели инструментов также является довольно быстрым (хотя и медленнее, чем использование горячих клавиш). Однако эти два способа не позволяют уточнить и изменить при необходимости параметры выполнения команды.

3. Выполнение команд с помощью меню – самый громоздкий и медленный способ действий, но при этом можно точно установить параметры выполнения команды.

В дальнейшем будут использоваться обозначения:

- названия пунктов меню и подменю, диалоговых окон будут приводиться в кавычках;
- названия виртуальных кнопок и клавиш (в диалоговых окнах) будут выделяться квадратными скобками;
- названия клавиш клавиатуры компьютера будут выделяться угловыми скобками: <…>.

В данном лекционном курсе для каждого действия приводится только один из возможных вариантов его выполнения.

2.2.2. Действия мышью

1. *Щелчок* – указатель мыши наводится на объект, нажимается и отпускается левая кнопка мыши. Щелчок чаще всего служит для выделения отдельных объектов, выполнения команд меню и "нажатия" виртуальных кнопок и клавиш;

2. *Двойной щелчок* – то же самое, но левая кнопка быстро нажимается два раза подряд. Служит для открытия папок и запуска программ;

3. *Щелчок правой кнопкой* – служит для вызова контекстного меню;

4. *Проводка* – перемещение мыши при нажатой левой кнопке. В основном применяется для выделения групп объектов или протяжённых областей документа;

5. *Перетаскивание* – мышь сначала наводят на объект и выделяют его, затем нажимают левую кнопку и проводят мышь по экрану. При этом выделенный объект перемещается вслед за указателем мыши. Перетаскивать можно самые разные объекты – окна или их границы, эмблемы (пиктограммы) на рабочем столе, рисунки в документе и др.

2.2.3. Работа с меню

Меню - важнейший элемент интерфейса ОС Windows. Различают следующие виды меню**:**

• главное меню или меню кнопки "Пуск", появляющееся на экране при нажатии виртуальной кнопки [Пуск] в нижнем левом углу экрана. Служит для запуска программ, настройки системы, выключения компьютера и других действий;

• контекстное меню;

• меню окна программы-приложения, стандартизованное для большинства приложений WINDOWS;

• подчиненное меню (подменю, ниспадающее меню);

Меню окна программы-приложения обычно находится под строкой заголовка окна и обеспечивает доступ ко всем функциональным возможностям программы.

Ниспадающее меню появляется при выборе пункта горизонтального меню. Пункты ниспадающего меню могут соответствовать отдельным командам или быть заголовками внутренних *подменю* – в этом случае рядом с заголовком находится *знак списка* в виде черного треугольника.

Контекстные меню появляются на экране после щелчка правой кнопкой мыши на объекте. В контекстном меню отражаются операции, которые можно выполнять с данным объектом в текущей ситуации. Набор команд в каждом контекстном меню зависит от мес-

та расположения курсора мыши в момент нажатия правой кнопки.

2.2.4. Основные элементы окна

Окна бывают трех видов:

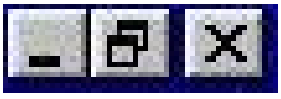
1. Окно программы – основное окно для работы с какой-либо программой;
2. Окно документа – расположено внутри окна программы и предназначено для работы с конкретным документом;
3. *Диалоговое* окно - окно, появляющееся при выполнении некоторых команд. Оно служит для уточнения параметров команды и может содержать следующие элементы:

Окна ввода для ввода каких-то пояснений с помощью клавиатуры или мыши; для ввода информации с клавиатуры в окно надо щелчком установить *курсор* – метку в виде мигающей вертикальной черты. Курсор является общим элементом интерфейса ОС Windows, его наличие всегда указывает на возможность ввода в данном месте информации с клавиатуры (или из буфера обмена). Для установки курсора надо щелкнуть мышью в нужном месте документа или окна;

- небольшие квадратные окошки, в которые щелчком мыши можно поставить **флажки –** метки в виде галочки. Такая метка означает, что помеченное действие или указание будет выполняться, отсутствие флажка – что указание не выполняется;
- наборы из двух или более небольших круглых окошек, в одном из которых установлен **переключатель** – метка в виде точки, которую можно переставлять щелчком мыши. Переключатель служит для выбора одного варианта из какого-то набора взаимоисключающих параметров команды.

После установки всех необходимых параметров следует нажать имеющуюся в окне клавишу [OK] или [Выполнить]. При отказе от выполнения команды следует нажать [Отмена] или просто закрыть окно имеющейся в нем кнопкой [×] – [Закрыть].

Для управления окнами предназначены одна (в диалоговых окнах) или три кнопки



Первая ([Свернуть]) убирает с экрана окно программы-приложения или документа и создает взамен виртуальную клавишу с

именем свернутого окна. При этом работа программы не прекращается. Свернутое окно в любой момент может быть возвращено на экран щелчком по упомянутой клавише.

Вторая ([Развернуть] или [Восстановить]) управляет размерами окна, позволяя сделать их максимально возможными (на весь экран) или уменьшить.

Третья ([Закрыть]) прекращает работу программы и убирает с экрана ее окно.

2.2.5. Панели инструментов

Панели инструментов приложений ОС Windows обычно располагаются в верхней части окна под строкой меню. Их можно выводить на экран или убирать с экрана по желанию пользователя. Список всех существующих в окне панелей инструментов можно вывести на экран с помощью меню (раздел "Вид" - команда "Панели инструментов") или вызвав контекстное меню щелчком правой кнопки мыши по одной из находящихся на экране панелей. При желании также можно изменить набор кнопок любой панели инструментов.

2.2.6.Буфер обмена

В Windows существует возможность обмена информацией между различными приложениями с помощью буфера обмена.

Буфер обмена **-** область памяти, в которую временно помещается объект или фрагмент документа.

Помещение в буфер какой-то информации производится командами "Скопировать" (в буфер записывается копия объекта) или "Вырезать" (в буфер переносится сам объект).

Команда "Вставить" создает копию содержимого буфера в том месте документа, в котором в данный момент находится курсор.

Команды работы с буфером имеются в любом приложении ОС Windows и в подпрограммах самой ОС. Они находятся в разделе "Правка" главного меню, в контекстных меню многих объектов, дублируются кнопками "стандартной" панели инструментов и горячими клавишами.

Записанный в буфер фрагмент можно вставить либо в другое место того же документа, либо в другой документ того же приложения, либо в документ другого приложения. Фрагмент, находящийся в

буфере обмена, может вставляться произвольное число раз.

Объект или фрагмент сохраняется в буфере обмена до тех пор, пока не поступила команда поместить в буфер новую порцию данных. В этом случае прежнее содержимое буфера обмена теряется безвозвратно и замещается новой информацией. При выходе из Windows содержимое буфера обмена также теряется.

2.2.7. Рабочий стол и его элементы

После загрузки Windows большую часть экрана занимает так называемый *рабочий стол*. На нем находятся:

1. *Пиктограммы* - графические эмблемы с подписями, соответствующие отдельным программам, папкам, файлам или устройствам компьютера. В зависимости от вида объекта двойной щелчок по его пиктограмме:

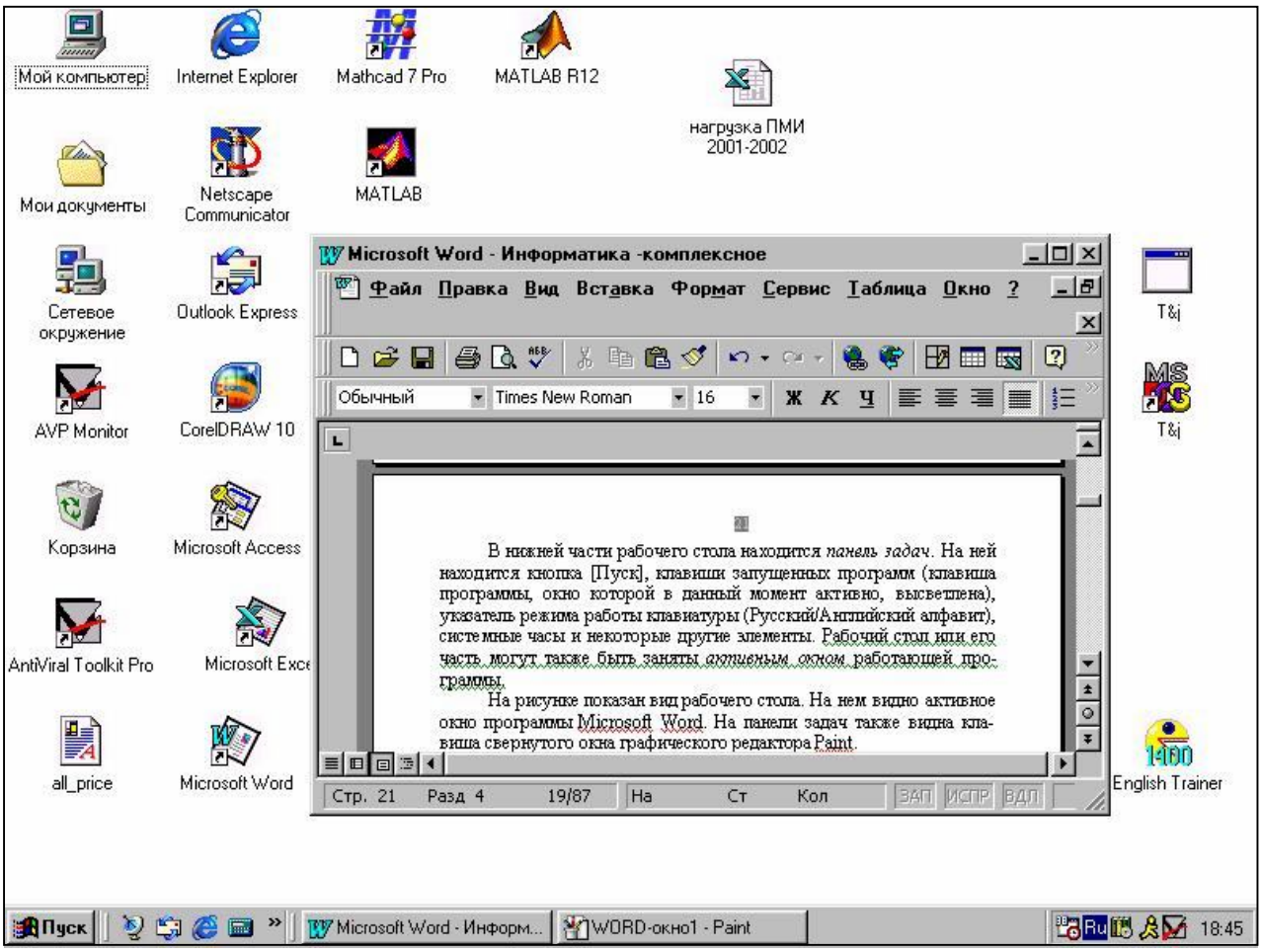
– Запускает программу;

– Открывает папку;

– Открывает диалоговое окно для настройки устройства.

В нижней части рабочего стола находится *панель задач*. На ней находится кнопка [Пуск], клавиши запущенных программ (клавиша программы, окно которой в данный момент активно, высветлена), указатель режима работы клавиатуры (Русский/Английский алфавит), системные часы и некоторые другие элементы. Рабочий стол или его часть могут также быть заняты *активным окном* работающей программы.

На рисунке показан вид рабочего стола. На нем видно окно программы Microsoft Word. На панели задач также видна клавиша свернутого окна графического редактора Paint.



Среди пиктограмм рабочего стола особое место занимают пиктограммы *ярлыков*, отмеченные небольшой стрелкой в нижнем углу. Ярлык – особый файл, содержащий информацию о местонахождении объекта (папки, файла, программы), которому он соответствует. Щелчок по пиктограмме ярлыка вызывает те же действия, что и щелчок по пиктограмме объекта. Однако уничтожение ярлыка не влечет за собой удаления самого объекта, поэтому на рабочий стол следует выносить пиктограммы именно ярлыков во избежание случайной потери нужной информации.

Для создания ярлыка следует

1. Щелкнуть левой кнопкой по рабочему столу и в контекстном меню навести мышь на значок списка пункта "Создать". В выпавшем подменю щелкнуть: "Ярлык";

2. В окно ввода открывшегося диалогового окна ввести имя файла, для которого создается ярлык (щелчком поставив в окно ввода

курсор) или найти этот файл в дереве папок, щелкнув клавишу [Обзор];

3. Щелкнув в диалоговом окне клавишу [Далее], отредактировать в следующем диалоговом окне имя ярлыка и щелкнуть клавишу [Готово].

2.2.8. Проводник.

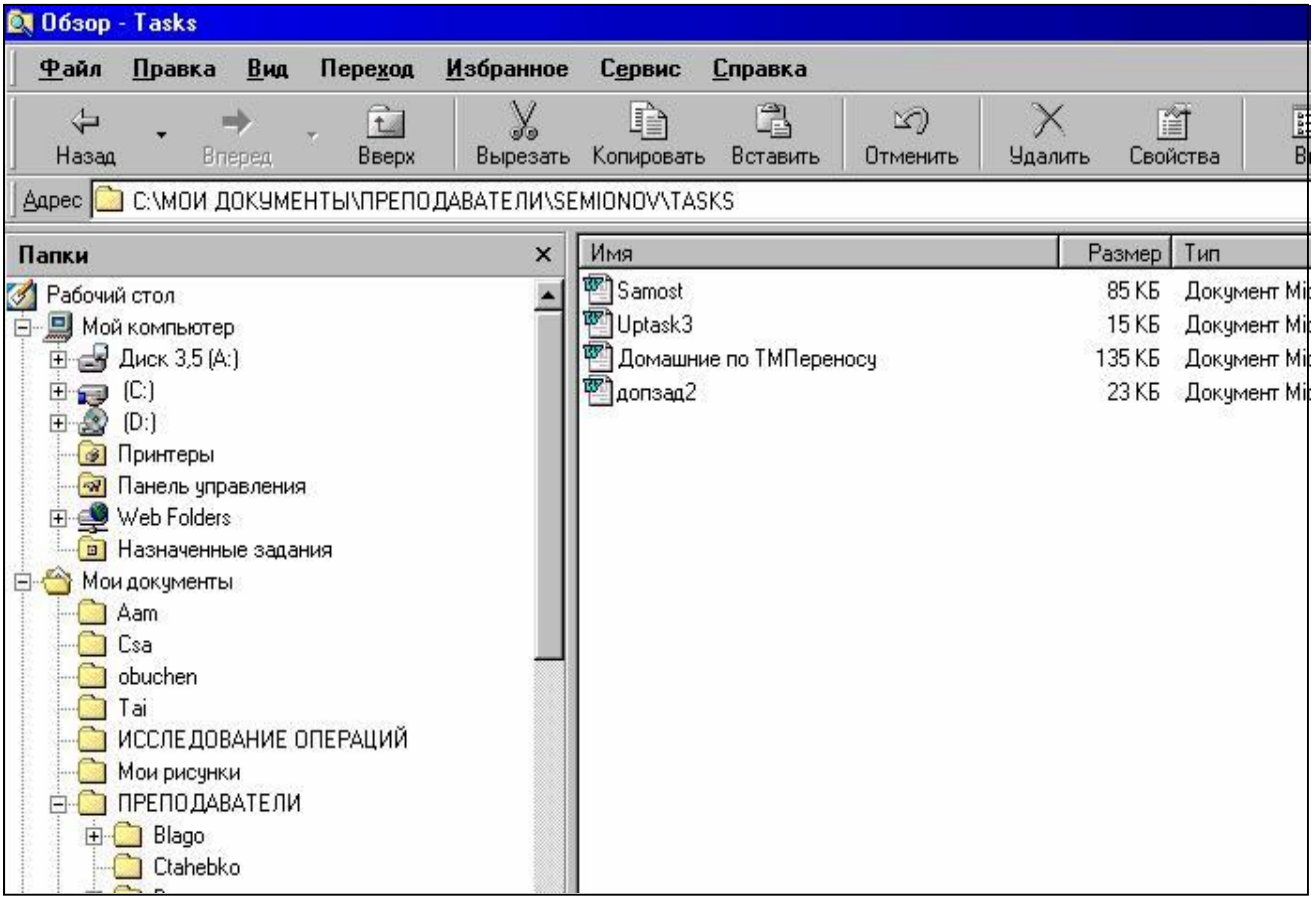
Проводник – подпрограмма ОС Windows, позволяющая производить различные действия с файлами (подобные программы называют *файловыми менеджерами*). Запуск Проводника осуществляется:

1. Через главное меню (щелкнуть по кнопке "Пуск", далее в главном меню и подменю последовательно выбрать пункты: "Программы" – "Проводник");
2. Двойным щелчком по ярлыку Проводника (если его пиктограмма есть на рабочем столе).

Окно программы Проводник (сверху вниз) состоит из:

1. строки заголовка (с кнопками управления окном в правой части);
2. строки меню;
3. панели инструментов;
4. строки адреса, в которой указывается имя активной папки;
5. двух панелей. На левой представлена структура дерева папок, а на правой — содержимое *активной* (выбранной на левой панели щелчком) папки. Папка Рабочий стол является главной (или родительской) по отношению к остальным в дереве папок.
6. Строки *статуса*, в которой дается некоторая информация о ходе работы.

На левой панели можно увидеть не только название папок, но и оценить их структуру: если папка содержит вложенные папки, то она помечается знаком "плюс".



Для раскрытия списка вложенных папок надо щелкнуть мышью по знаку "плюс", папка откроется и отобразится ее структура, а знак "плюс" изменится на знак "минус" который указывает на то, что папка открыта, т.е. видно ее содержание. Для свертывания папки нужно мышью щелкнуть по знаку "минус".

Вид правой панели можно изменить с помощью меню ("Вид" – " Крупные значки", или "Мелкие значки", "Список", "Таблица").

Основные приемы работы с файлами:

1. *Поиск* файла/папки:

А) щелкнуть правой кнопкой на левой панели папку, в которой предполагается найти объект и в контекстном меню выбрать "Найти…" (вариант – выбрать пункт меню "Поиск" затем в подменю: "Файлы и папки");

Б) указать в окнах ввода диалогового окна поиска имя объекта, имеющийся в нем текстовый фрагмент и уточнить, если надо, папку, в которой следует искать. Затем щелкнуть [Найти].

2. *Копирование* файла/папки:

А) активизировать папку, содержащую исходный файл или вложенную папку;

Б) пометить щелчком копируемый объект;

В) скопировать объект в буфер (командой "Копировать" из меню "Правка" или контекстного меню помеченного объекта, или кнопкой панели инструментов);

Г) пометить слева папку, в которую копируется объект и произвести вставку из буфера командой "Вставить" (выполняется теми же способами, что и "Копировать").

3. *Перемещение* файла/папки.

Выполняется так же, как и копирование, только вместо команды "Копировать" используется команда "Вырезать".

4. *Переименование* файла/папки:

А) пометить объект (на любой панели);

Б) выполнить команду "Переименовать" (через раздел меню "Файл" или контекстное меню объекта). Рядом с именем объекта начинает мигать курсор.

В) изменить имя объекта и нажать <Enter>.

5. *Создание* новой папки.

А) активизировать папку, в которую будет вложена созданная;

Б) выполнить команду "Создать" (через раздел меню "Файл" или контекстное меню правой панели, вызываемое щелчком правой кнопки мыши по полю правой панели).

В) в появившемся подменю выбрать пункт "Папку". На правой панели появляется пиктограмма папки с условным именем "Новая папка" и курсором. Удалить условное имя и набрать на клавиатуре нужное имя для новой папки, затем нажать <Enter>.

6. *Удаление* файла/папки.

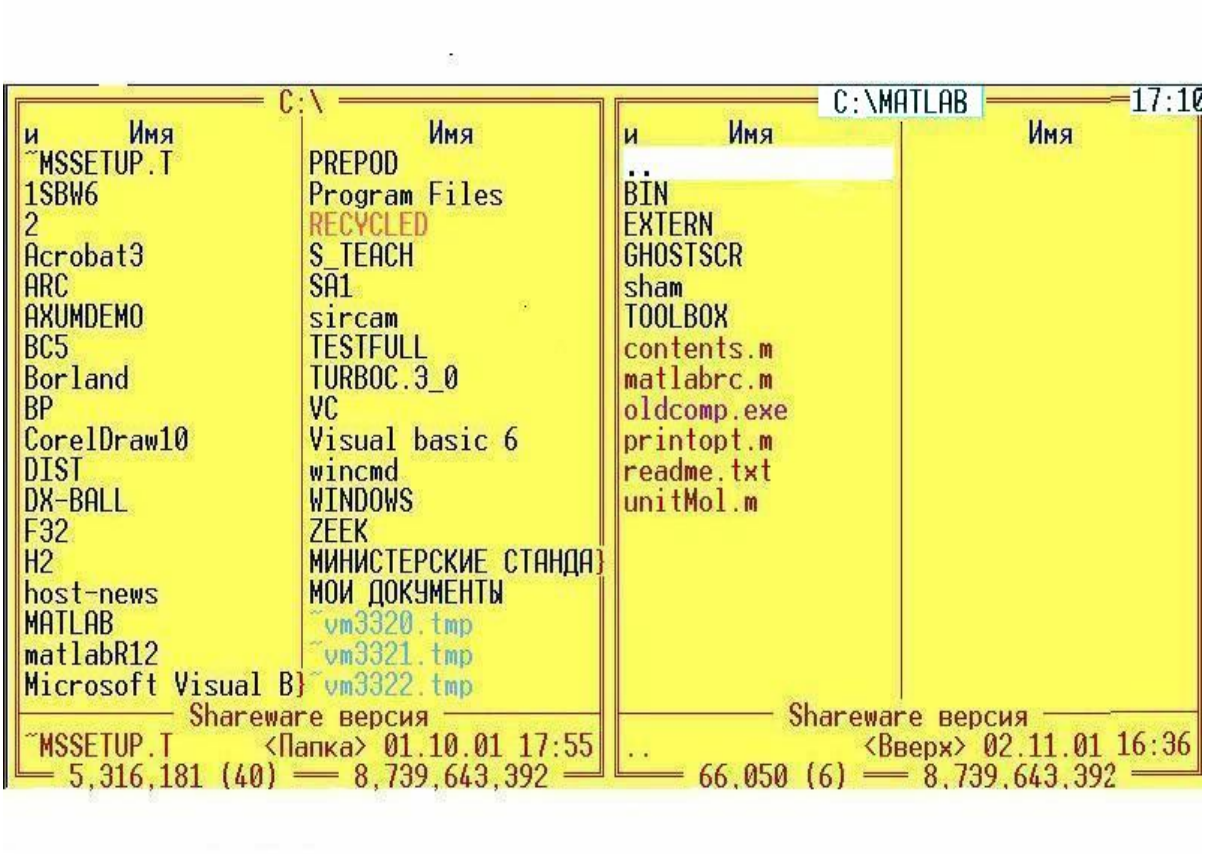
А) пометить объект;

Б) выполнить команду "Удалить" (из раздела меню "Файл" или контекстного меню объекта). Подтвердить необходимость удаления в ответ на запрос компьютера.

После этого объект перемещается в особую папку винчестера, называемую "Корзина". Физическое удаление объекта с винчестера происходит при очистке корзины соответствующей командой из ее контекстного меню. Пока очистка не произведена, объект можно восстановить на старое место, открыв корзину, как обычную папку, пометив объект на правой панели и выполнив команду "Восстановить" из контекстного меню объекта. При необходимости восстановления объекта на другое место это можно сделать, перенеся объект на это место, как описано выше.

2.2.9. Файловый менеджер FAR.

Еще до появления ОС Windows для операционной системы MS DOS была разработана программа-надстройка *Norton Commander (NC)*. Принцип организации этой надстройки оказался настолько удачен, что после появления ОС Windows для нее также была создана программа *FAR Manager*, в основе своей повторяющая структуру NC.



После запуска FAR на экране появляется окно. Оно содержит две панели, каждая из которых раскрывает содержание какой-то папки. Имя раскрытой папки находится в верхней части панели. Если папка является вложенной в некую внешнюю, список содержимого начинается строкой с двумя точками. Одна из строк одной из панелей выделена цветным фоном (активизирована). Переведя фоновую метку на строку с двумя точками и нажав <Enter>, можно выйти в папку более высокого уровня, выделив строку с именем вложенной папки и нажав <Enter> - войти во вложенную папку.

Перемещение выделяющей метки производится клавишами управления курсором на клавиатуре компьютера (клавиши со стрелками, <Page Up>, <Page Down>). Переход на другую панель – нажатием клавиши <Tab>. Для перехода в другой корневой каталог при-

меняется "горячая" комбинация клавиш <Alt> + <F1> для левой панели и <Alt> + <F2> для правой.

Если какие-то действия надо провести с группой файлов, их помечают, поочередно наводя на них метку и нажимая клавишу <Insert>. Повторное нажатие той же клавиши снимает пометку.

Для запуска программы надо найти и выделить на любой панели файл, содержащий данную программу и нажать <Enter>. Непосредственный запуск возможен для файлов, содержащих программы в машинных кодах (с расширением *exe*).

Основные операции с файлами производятся с помощью функциональных клавиш:

F3 – просмотр содержимого файла без его редактирования или запуска содержащейся в нем программы; так можно просматривать файлы, содержащие данные в текстовом формате;

F4 – редактирование содержимого текстового файла;

F5 – копирование файла. Копирование проводится по принципу "с панели на панель": надо на одной из панелей раскрыть каталог, в котором будет создана копия файла, а на другой найти и выделить файл, который следует скопировать (или пометить группу файлов). Затем нажимается клавиша <F5>, после чего следует дополнительный запрос компьютера с указанием имени копируемого файла и каталога, в который идет копирование. При необходимости можно внести уточнения, затем нажать <Enter> или щелкнуть мышью заголовок "Copy" в нижней части окна запроса. Для отмены копирования надо нажать <Esc> или щелкнуть заголовок "Cancel" ("Отменить") в нижней части окна.

F6 – переименование или перенос файла. Нажатие клавиши вызывает окно, в котором по умолчанию указано имя папки, открытой на соседней панели и предлагается перенос выделенного файла в эту папку. Если это и есть цель нажатия клавиши <F6>, достаточно просто нажать <Enter>, в противном случае в строке ввода надо указать новое имя для файла (с маршрутом).

F7 – создание новой папки. При нажатии клавиши открывается окно, в строке ввода которого надо указать имя создаваемой новой папки. Новая папка будет вложена в ту папку, которая была раскрыта на панели, один из файлов которой был выделен в момент нажатия клавиши <F7>.

F8 – удаление выделенного файла или помеченной группы файлов.

Назначение других функциональных клавиш:

F1 – вызов справочной системы;

F2 – вызов и редактирование пользовательского меню для быстрого запуска часто используемых программ и отдельных файлов;

F9 – вызов верхнего меню FAR, служащего для настройки окна и выполнения некоторых команд, в частности, поиска файлов.

F10 – закрытие окна и окончание работы FAR.

2.3. СИСТЕМА ПРОГРАММИРОВАНИЯ TURBO PASCAL

2.3.1. Основные характеристики языка Pascal

Паскаль – язык программирования высокого уровня, созданный швейцарским программистом Н. Виртом первоначально как язык для обучения студентов программированию. Будучи учебным языком, Паскаль имеет очень ясную структуру, облегчающую его изучение и написание программ. Эти качества оказались настолько привлекательными, что Паскаль постепенно получил применение в качестве языка практического программирования, хотя и имеет ряд недостатков, обусловленных первоначальной направленностью только на обучение.

Изучение языка Паскаль имеет значение не только в плане овладения понятиями и приемами программирования, но еще и потому, что на его базе строится одна из наиболее популярных сегодня систем программирования Delphi.

Для программирования на языке Паскаль используется система TurboPascal, разработанная американской фирмой Borland.

Программы, написанные на языке Паскаль, допускают операции со следующими основными типами данных:

1. Целые числа в диапазоне –32768 ... +32767: тип *Integer*. Существуют дополнительные типы целых чисел, различающиеся диапазоном значений и объемом памяти, требующимся для записи одного числа: *Byte, Word, Shortint, Longint*. Например, число типа *Byte* записывается с помощью 1 байта памяти и может лежать в диапазоне 0 … 255. Для записи числа типа *Integer* нужно 2 байта.

2. Вещественные числа: тип *Real*. Для вещественных чисел также существуют дополнительные типы, но они используются достаточно редко.

3. Логические значения: тип *Boolean*. Логическое значение подразумевает ответ на вопрос "Верно ли, что …?" и может быть одним из двух: *True* ("Да" или "Истинно") и *False* ("Нет" или "Ложно").

4. Символы (таблицы ASCII и ее расширения): тип *Char*.

5. Строки – наборы символов: тип *String*.

6. Записи (тип *Record)* – наборы значений различных вышеперечисленных типов, объединенных под общим именем. Отдельные значения ("поля") внутри записи различаются с помощью внутренних имен.

7. Массивы (тип *Array*) – наборы значений любого из вышеперечисленных типов (в том числе массивы записей), объединенных под общим именем и различаемых внутри массива по номерам (индексам). В зависимости от числа индексов массивы могут быть *одномерными* и *двумерными* (индексов может быть и больше).

Существуют и более сложные типы данных.

Для обработки данных в языке Паскаль используются следующие управляющие структуры:

1. Простые *операторы* – отдельные команды языка.
2. Составные операторы или *блоки* – наборы операторов, объединенные так называемыми *операторными скобками* – служебными словами *Begin* ("Начало") и *End* ("Конец").
3. Подпрограммы – наборы операторов, имеющие собственное имя и описывающие обобщенно последовательность действий по обработке каких-то условных объектов. В программе подпрограммы обычно применяются несколько раз в разных местах или для обработки разных реальных объектов программы.
4. *Модули* – наборы подпрограмм, записанные по особым правилам безотносительно к конкретной программе. Подпрограммы, включенные в модуль, могут применяться в различных программах, если модуль *подключить* к программе особой декларацией.

Алфавит языка Паскаль включает в себя:

1. Заглавные и строчные латинские буквы;
2. Цифры 0, 1 , … , 9;
3. Знаки математических операций и отношений +, -, *, / и отношений =, <, >;
4. Знаки препинания : . , ; : , символ подчеркивания _ , апостроф ' ;
5. Скобки ( ), [ ], { };

Буквы русского алфавита допустимо использовать только в качестве символьных переменных.

Каждый объект, использованный в программе (переменная, константа, тип и др.), должен иметь имя. Имена большинства объектов могут иметь произвольную длину, но должны состоять только из букв, цифр и символов подчеркивания (пробелы не допускаются), причем начинаться обязательно с буквы. Исключением являются имена *меток*, которые могут состоять и просто из цифр.

Имена объектов не должны совпадать со служебными словами Паскаля.

2.3.2. Среда программирования TurboPascal.

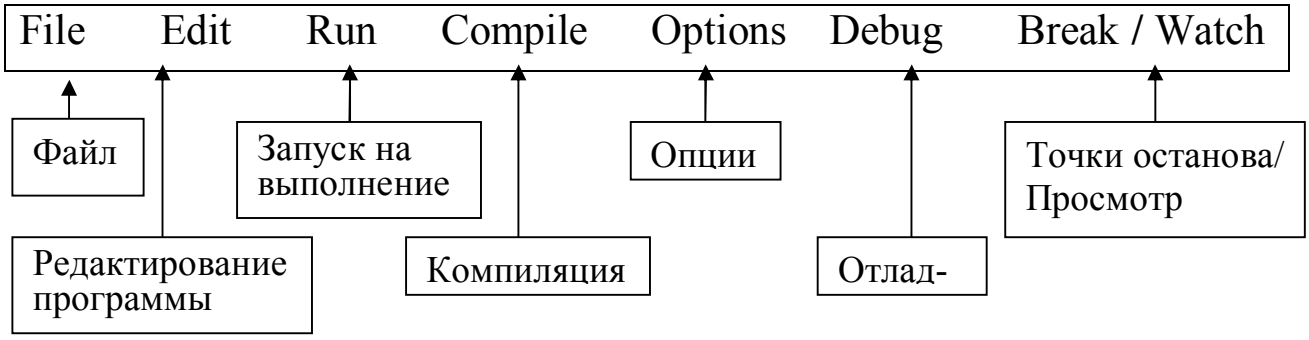
Среда программирования включается запуском файла *Turbo.exe.* После запуска на выполнение файла Turbo.exe программист видит перед собой экран, содержащий:

- ***Главное меню*** в верхней строке. Это перечень функций, выполняемых системой, команд и настроек.
- ***Окно редактирования Edit –*** рабочее пространство для работы с текстом программы.
- ***Окно отладки Watch –*** в нём размещается специфическая информация, необходимая при отладке программ (может быть скрыто).
- ***Строка-подсказка***, дающая расшифровку действий функциональных клавиш ***F1-F10.***

Имеется еще окно просмотра результатов, которое появляется на экране во время выполнения программы и исчезает немедленно после ее окончания. Переход из окна редактирования в окно просмотра и обратно совершается нажатием "горячей" комбинации клавиш <Alt> + <F5>.

Отредактированную программу можно запустить командой *Run* из одноименного раздела главного меню, или "горячей" комбинацией клавиш <Ctrl> + <F9>.

Разделы главного меню имеют следующие названия:



Активизация меню осуществляется клавишей <F10>. Каждое слово меню (кроме *Edit*) имеет вертикальное подменю, которое активизируется нажатием клавиши ввода (Enter). Перемещение светящегося фона (выделения) по пунктам меню и подменю осуществляется клавишами со стрелками. Активизировать требуемый пункт меню можно также через заглавную букву соответствующего слова: F, E, R, C, O, D, B.

Пункт меню *File* имеет подменю:

Load F3	Чтение файла
Pick Alt+F3	История работы
New	Новый файл
Save F2	Запись файла
Write to	Запись файла с новым именем
Directory	Просмотр каталога
Change Dir	Смена каталога
Dos Shell	Временный выход в MS-DOS
Quit Alt+X	Выход из системы

Активизация пунктов подменю осуществляется перемещением светящегося фонового выделения и нажатием клавиши <Enter>.

При активизации пункта *Load* возникает окно, в строке ввода которого имеется "маска" файла: * . * . Символы * заменяют в маске конкретные имена и расширения файлов. Следует стереть маску и ввести полное (со всеми атрибутами) имя файла программы, который надо прочитать с какого-либо носителя. Можно просто нажать <Enter> и открывается окно с полным списком файлов текущего каталога (в котором находится файл *Turbo.exe*). Если удалить в маске

только второй символ * и ввести вместо него конкретное расширение (например, *.pas), будет выведен список тех файлов, которые имеют такое расширение. Правила перемещения по дереву каталогов с помощью окна такие же, как в программе FAR Manager, только строка с двумя точками, обозначающая выход во внешний каталог, находится в конце, а не в начале списка файлов. Надо войти в нужную папку, выделить имя нужного файла и нажать <Enter>. Файл читается с носителя и текст программы помещается в окно *Edit*. При включении окна *Edit* в нём появляется мигающий курсор, а в верхней строке рабочая информация:

Line **...** – № текущей строки;

Col **…** -- № текущего столбца;

Insert – показывает, что установлен режим вставки при вводе символов;

Indent – показывает, что установлен режим автоматического отступа: после перевода строки начало новой строки автоматически подгоняется под начало предыдущей;

Tab – говорит о том, что при нажатии клавиши-стрелочки влево и вправо (первая клавиша во втором ряду основного поля) в текст будет автоматически вставлено несколько (по умолчанию до 8) пробелов или их удаление.

Fill, Unindent – режимы замены …(пользоваться не будем).

C: NONAME. PAS – поставленное по умолчанию условное имя текущего файла (файла, с которым производится работа).

Набор и редактирование текста программы производятся по следующим правилам:

- разница между заглавными и строчными буквами не имеет значения;
- отдельные слова и имена должны отделяться от соседних как минимум одним пробелом или знаком препинания, математического действия. Наличие большего количества пробелов не имеет значения;
- наличие пустых строк не имеет значения; пустыми строками текст программы можно визуально разделить на отдельные структурно значимые фрагменты;
- нажатие клавиши <Enter> вызывает перевод курсора в следующую строку;
- любой набор символов, заключенный между фигурными скобками {…}, называется *комментарием*. Комментарий не влияет на исполнение программы.

Набранный текст программы следует сохранить, записав в виде файла на каком-либо носителе. Для сохранения файлов используются команды раздела меню *File : Save* (Сохранить) и *Write to* (Сохранить как). При первоначальном сохранении файла используется команда *Write to*. При ее выполнении открывается окно ввода, в котором следует набрать с клавиатуры имя нового файла с указанием только расширения при сохранении в текущем каталоге или маршрута и расширения при сохранении в каком-либо другом каталоге; затем нажать <Enter>. Записать файл в другой каталог можно также иным способом. Для этого следует после выполнения команды *Write to* и появления окна ввода вторично нажать <Enter>. Открывается окно со списком файлов текущего каталога. Надо перейти в каталог, в котором будет записан файл, затем нажать <Esc> (возврат в подменю *File* с выделением команды *Write to*) и снова <Enter>. Опять возникает окно ввода, но в нем уже записан маршрут для нужного каталога, так что надо только дополнить его именем и расширением файла.

Команда *Save* применяется для записи нового варианта файла под старым именем на старое место.

2.3.3. Общая структура программы на языке Паскаль

Программа на Паскале имеет следующую общую структуру:

PROGRAM имя_программы; {заголовок программы}
USES список используемых модулей; {сообщение о включении модулей}
LABEL список меток; {объявление меток}
CONST константа = значение; {объявление констант}
TYPE имя типа=описание; {определение типов данных}
VAR имя: тип; {объявление глобальных переменных}
PROCEDURE имя; {объявление подпрограмм-процедур}
FUNCTION имя; {объявление подпрограмм-функций}
BEGIN {начало тела программы}
…
(Операторная часть программы;)
…
END. {конец программы}

Наличие каждой из пяти секций объявлений - USES, LABEL, CONST, TYPE, VAR, PROCEDURE и FUNCTION в программе необязательно и определяется ее структурой.

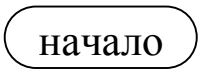
2.3.4. Алгоритм. Описание алгоритма. Блок-схемы.

Написанию программы предшествует разработка *алгоритма*. Алгоритм – это однозначно определенная последовательность конечного числа шагов, необходимых для достижения цели. Алгоритмы бывают:

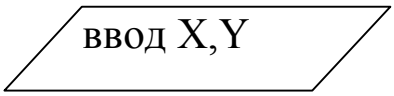
1. *Линейные*, когда шаги выполняются поочередно один за другим без пропусков или повторений;
2. *Разветвляющиеся*, когда в зависимости от выполнения каких-то условий может выполняться та или иная последовательность шагов;
3. *Циклические*, когда какая-то последовательность шагов повторяется несколько раз.

Отдельные типы алгоритмов могут сочетаться друг с другом.

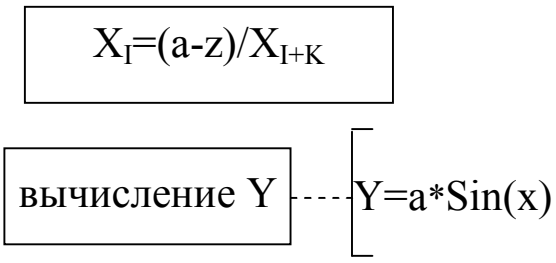
Описание алгоритма может быть словесным но часто для описания структуры алгоритма применяется язык *блок-схем*. В этом случае отдельные шаги алгоритма изображаются условными графическими элементами.



Элемент пуска-останова определяет начало или конец процесса выполнения программы. Если элемент соответствует началу, то внутри записывается слово ‘начало’, для конца - слово ‘конец’.



Элемент ввода-вывода соответствует вводу исходных данных решаемой задачи и выводу результатов. Для ввода в него записывается слово ‘ ввод’ и перечисляются переменные, значения которых подлежат вводу, т.е. записывается список ввода. При выводе - слово ‘вывод’ или ‘печать’ и список вывода.



Элемент вычисления или присваивания значения содержит внутри запись формулы или слово ‘вычисление’, а далее имя переменной, являющейся результатом выполнения данного оператора, рядом в виде комментария записывается формула.

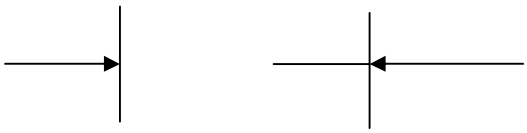
Элемент выбора дальнейшего направления решения (разветвления). Выбор осуществляется в зависимости от выполнения условия, записанного внутри элемента. Если условие выполняется (ответом на вопрос условия является слово ‘да’), то продолжение вычислительного процесса осуществляется по ветке ‘да’. В противном случае - по ветке ‘нет’.

Прочие элементы блок-схем будут рассмотрены впоследствии.

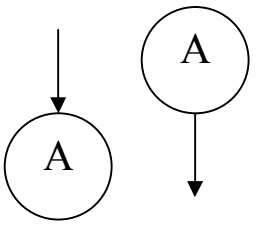
Блоки соединяются в схему горизонтальными, или вертикальными линиями, или ломаными, состоящими из горизонтальных и вертикальных отрезков. Эти линии называются *линиями потока*.

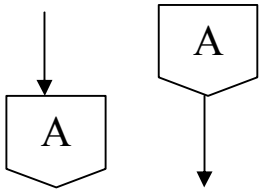
Направления линий потока слева направо и сверху вниз считаются естественными. Противоположные направления - неестественными, их окончание положено фиксировать стрелочками.

Линии потока могут пересекаться и образовывать слияние. Место слияния фиксируется точкой.



Часто из-за пересечения линий потока блок-схемы теряют наглядность. При такой ситуации лучше их разрывать, а для условного соединения пользоваться соединителями, расположенными в начале и в конце разрыва линии потока.





Разрывы изображаются фигурами: внутристраничный – кругом, межстраничный – "открытым конвертом". Внутри фигур, относящихся к одной и той же линии, записываются одинаковые метки: буквы, числа, символы или их сочетания.

В межстраничных соединителях можно указывать две метки, взяв в качестве второй для соединителя, расположенного в начале разрыва, номер страницы, на которой следует искать продолжение данной линии, а для соединителя в конце - номер страницы, на которой надо искать начало данной линии.

При желании все блоки схемы можно пронумеровать сквозными номерами, проставив их слева в разрывах верхних линий. Блоки начала и конца не нумеруются.

2.3.5. Раздел описаний

Раздел описаний располагается между второй строкой про-

граммы (строкой подключения модулей) и ключевым словом *Begin*, открывающим *раздел операторов*. В разделе описаний должны быть перечислены все использованные в программе имена объектов с указанием вида объекта (метка, константа, пользовательский тип, переменная, подпрограмма), а для переменных – тип значения каждой переменной. Тип константы определяется автоматически при задании ее значения.

Описание констант и переменных заставляет компьютер произвести распределение ячеек памяти для записи их значений, однако начальные значения переменных при этом не задаются. Это должно быть сделано в разделе операторов с помощью присваивания значений, вычисления, ввода с клавиатуры или чтения из отдельного файла данных.

Примеры описаний:

```
Label START, 1,1000; {описание меток}
Const     E = 2.81828;{описание вещественной константы}
          KONEC = 'Ура!'; {описание константы - строки}
Type      MATRIX = array [1 .. 3, 1 .. 5] of real; {описание типа
                                  двумерного массива вещественных чисел}
Var       I, J, K : integer; {описание целых переменных}
          A, B, R : real;{описание вещественных переменных}
          TABL : MATRIX; {описание массива типа MATRIX}
          TABLICA: array [1 .. 3, 1 .. 5] of real; {описание массива той же структуры, как
                                  переменной}
          RJAD: array [1 .. 10] of integer; {одномерный массив
                                  целых чисел}
          BUKVA: char; {описание символьной переменной}
          SLOVO: string [20]; {описание строки из 20 символов}
          FRAZA: string; {описание строки максимальной длины: 127 или 255 символов в разных
                                  версиях TurboPascal }
```

2.3.6. Простые операторы

1. Оператор *присваивания*

<имя простой переменной>:=<выражение>;

В результате выполнения в ячейку памяти, соответствующую данной переменной, заносится значение выражения. Выражением может служить имя другой переменной, или формула для вычисления, в том числе с использованием предыдущего значения этой же переменной, например:

```
J:=J+1;
```

При выполнении этого оператора в ячейку памяти, содержащую значение переменной J, заносится новое значение, увеличенное на 1.

Выражение состоит из операндов и операций. Операнды – это отдельные слагаемые, сомножители и т.п., операции – действия над ними.

В Паскале предусмотрены арифметические операции:

\+ сложение,

\- вычитание,

* умножение,

/ деление,

DIV – целочисленное деление (деление целых чисел нацело с отбрасыванием остатка), например: **5 div 2 = 2**

MOD – определение целочисленного остатка от деления нацело, например: **5 mod 2 = 1.**

Операция возведения в степень в Паскале отсутствует. Для вычисления значения X^A используется выражение exp(A*ln(X)) .

Математические функции, предусмотренные в Паскале (X – вещественное число):

abs(X) – вычисление абсолютной величины аргумента X;

arctan(X) – арктангенс X;

cos(X) – косинус X;

exp(X) – экспоненциальная функция;

frac(X) –дробная часть числа;

int(X) –целая часть вещественного числа (записанная в памяти, как вещественное число);

ln(X) – логарифм X

Pi – функция без аргумента, вычисляющая значение числа π;

Sin(X) – синус X;

round(X)- округление вещественного до целого (результат имеет тип integer);

trunc(X) – усечение вещественного путем отбрасывания дробной части (результат типа integer);

sqrt(X) – квадратный корень числа

sqr(X) –квадрат числа (аргумент может быть и целым, тогда результат будет тоже целым).

2. Оператор *перехода*

goto <имя метки>;

Результатом выполнения оператора является переход не к следующему по порядку оператору, а к оператору, перед которым стоит метка, например:

```
. . . . .
goto 1;
. . . . .
1: J:=J+1;
. . . . .
```

Все метки, имеющиеся в программе, должны быть описаны в разделе описаний (см. п. 2.3.5).

Использование операторов перехода, особенно в сложных программах, запутывает структуру программы, усложняет ее понимание и увеличивает вероятность ошибок. Поэтому использование операторов перехода "в чистом виде" не рекомендуется. В основном они применяются как составная часть условных операторов (см. далее п.2.3.8).

2.3.7. Процедуры ввода-вывода

Составными частями любого алгоритма являются ввод исходных данных и вывод на экран монитора или на печать полученных результатов. Для этого в Паскале предусмотрен ряд процедур.

1) Процедуры ввода

Read (<список ввода>);

Readln (<список ввода>);

Список ввода – это перечень имен простых переменных или отдельных элементов массивов. Элементы списка перечисляются через запятую, а во время ввода их значений с клавиатуры отдельные значения отделяются друг от друга пробелами.

2) Процедуры вывода

Write(<список вывода>);

Writeln(<список вывода >);

Список вывода похож на список ввода, но в нем могут иметься еще строки – текстовые фрагменты или просто наборы символов, заключенные между двумя апострофами и служащие для придания наглядности выводимым на экран (печать) результатам. Для улучшения вида выводимых числовых значений предусмотрен *формат вывода*. Для целых чисел формат вывода – это указание количества позиций на экране, занимаемых данным числом. Оно ставится после имени выводимой переменной в списке вывода и отделяется от имени двоеточием. Если выводимое число окажется короче, то лишние позиции перед первой цифрой числа заполняются пробелами, если длиннее – количество позиций автоматически увеличивается.

Для вещественных чисел формат вывода состоит из двух целых чисел, также разделенных двоеточиями. Первое показывает общее число занимаемых на экране позиций, а второе – количество цифр дробной части числа. Если дробная часть окажется короче, или отсутствует, свободные позиции заполняются нулями, если длиннее – производится округление. Если указанной ширины поля недостаточно из-за длинной целой части, то выводится число в формате с плавающей точкой и 10 знаками после запятой. Если формат вывода вещественного числа не указан, оно выводится в форме с плавающей точкой.

Следует учесть, что при выводе элементов пробелы между ними автоматически не вставляются.

Две буквы ' ln ' (сокращение от английского "*line*" –"строка") в конце имени процедуры означают в обоих случаях, что после выполнения ввода или вывода курсор на экране автоматически переводится в следующую строку. При использовании процедур *read* и *write* курсор остается в той же строке.

Существуют *пустые* процедуры ввода-вывода, в которых отсутствуют списки:

Readln;

Writeln;

Первая вызывает приостановку выполнения программы до момента, когда на клавиатуре будет нажата клавиша ввода <Enter>. Вторая – вызывает перевод курсора на экране в начало следующей строки.

Пример: вывод на экран значений переменных J=12 и A=2.81828.

```
writeln ('J=' , J : 5 , 'A=' , A : 8 : 3);
```

Результат:

```
J=   12A=   2.818
```

Между последней цифрой значения J и началом текста строки 'A=' не оказалось пробела. Чтобы отделить на экране эти элементы, надо было в списке вывода задать строку ' A=', вставив в нее пробел перед буквой 'A'.

2.3.8. Пример программирования линейного вычислительного процесса

Задача. С клавиатуры вводятся значения длин сторон треугольника A, B, C. Необходимо вычислить площадь треугольника.

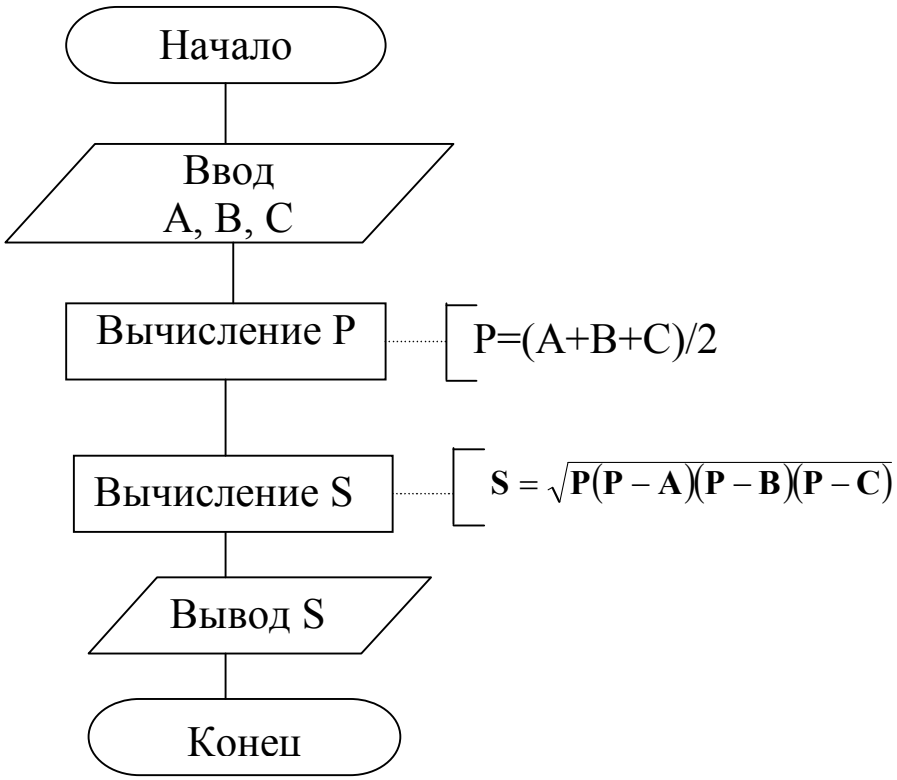
Постановка задачи. Для вычисления используем формулу Герона

$$\mathbf{S} = \sqrt{\mathbf{P}(\mathbf{P}-\mathbf{A})(\mathbf{P}-\mathbf{B})(\mathbf{P}-\mathbf{C})},$$

где **P** – полупериметр треугольника. Таким образом, при решении задачи используются следующие простые переменные вещественного типа:

1. A, B, C – длины сторон треугольника (исходные данные);
2. P = (A+B+C)/2 – полупериметр треугольника (вспомогательная переменная);
3. S – площадь треугольника.

Блок-схема алгоритма.



Текст программы.

```
Program PRIMER;
uses  CRT;              {подключаем стандартный модуль CRT }
      {Раздел описаний}
var A, B, C : real;     {исходные данные}
P : real;               {промежуточный результат}
S : real;               {конечный результат}
      {Основной блок}
Begin
  ClrScr; {процедура очистки экрана из модуля CRT }
  writeln ('Ввод исходных данных:');
  writeln ('Введите длины сторон, разделяя значения пробелами:');
  readln (a, b, c); {ввод исходных данных с предварительным
                     сообщением}
  P: = (A+B+C)/2;  {оператор присваивания – вычисление P}
  S: = sqrt(P*(P-A)*(P-B)*(P-C));
  writeln ('S = ' , s : 8 : 2, '  кв. см.'); {вывод S в поле из 8 позиций
                                              с 2 цифрами после точки и с
                                              пояснением}
```

readln; {пустой оператор для приостановки работы программы, дающий возможность увидеть результаты в окне просмотра, см. п. 2.3.7}
End. {конец программы}

2.3.9. Условный оператор и оператор выбора

Условный оператор позволяет осуществить разветвление вычислительного процесса. Различают полную и неполную форму условного оператора. Полный условный оператор имеет структуру

if <условие> then <оператор 1> else <оператор 2>;

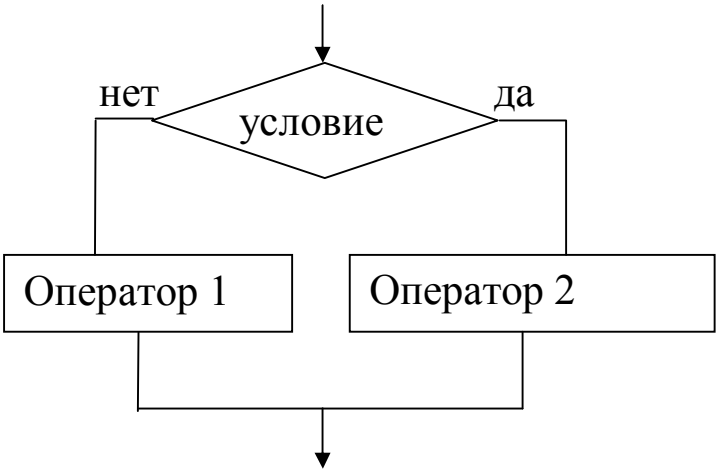
В случае выполнения условия в программе выполняется оператор 1, а если условие не выполнено, то выполняется оператор 2. После этого выполнение программы продолжается в порядке следования дальнейших операторов.

Неполный условный оператор имеет структуру

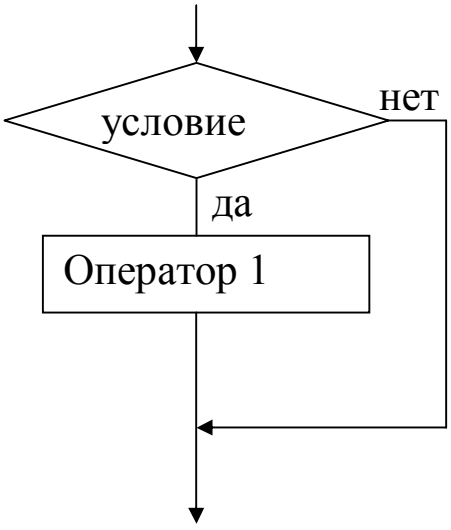
if <условие> then <оператор 1>;

Если условие выполнено, то выполняется оператор 1, а если условие не выполнено, то происходит переход к следующему по порядку оператору программы.

В блок-схемах алгоритмов полному и неполному условным операторам соответствуют фрагменты



полный



неполный

Любой из отдельных операторов в структуре условного оператора может снова быть условным оператором. Это позволяет проводить сложные ветвления вычислительного процесса.

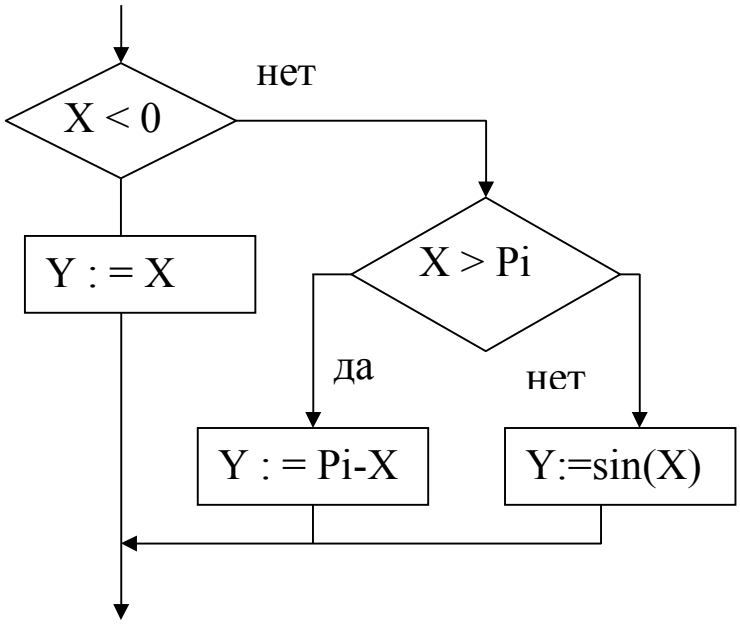
Пример: вычислить выражение

$$y = \begin{cases} x & \text{при } x < 0 \\ \sin x & \text{при } 0 \le x \le \pi \\ \pi - x & \text{при } x > \pi \end{cases}$$

Отметим, что второе условие является сложным и распадается на два простых: $x \ge 0$ и $x \le \pi$. Чтобы избежать записи сложного условия, можно использовать логическое *правило исключенного третьего*:

Если имеются два взаимно противоречащих друг другу условия, то какое-то одно из них всегда выполнено, второе – не выполнено, а третьего не существует.

Очевидно, что любое значение **x** либо отрицательно, либо неотрицательно. Если поставить условие **x < 0**, то его невыполнение автоматически означает, что **x ≥ 0**. В этом случае значение **y** должно вычисляться либо по второй формуле, либо по третьей, других возможностей уже не существует. Поэтому можно выбрать любое из оставшихся условий – при его выполнении следует использовать соответствующую ему формулу, а при невыполнении автоматически применить оставшуюся третью формулу, уже не ставя для нее никаких дополнительных условий. В качестве второго условия, таким образом, можно взять условие **x ≥ π,** записать которое легче. Соответствующий условный оператор имеет вид



```
if  X<0
   then
     Y:=X
   else
     if X>Pi
        then
          Y:=Pi - X
        else
          Y:=sin(X);
```

Из этого примера видно, что простые условия чаще всего имеют вид математических отношений (равенства или неравенства). Допускается постановка условий в виде отношения для символьных переменных, при этом фактически сравниваются между собой их номера в таблице кодов ASCII; в частности, правильными (выполняемыми) неравенствами будут

A < F, y < z, A < a .

Последнее неравенство верно, так как в таблице кодов ASCII заглавные буквы стоят раньше строчных.

В качестве простого условия можно также использовать значение логической переменной.

Даже пользуясь правилом исключенного третьего, не всегда удается избежать использования сложных условий. В этом случае для объединения простых условий используются специальные логические операции:

1. *Логическое сложение* (операция "или") записывается служебным словом *or ("или")*:

if <условие 1> or <условие 2> then . . .

Сложное условие в этом случае считается выполненным, если выполнено хотя бы одно из условий: 1 или 2.

2. *Логическое умножение* (операция "и") записывается служебным словом *and ("и")*:

if <условие 1> and <условие 2> then . . .

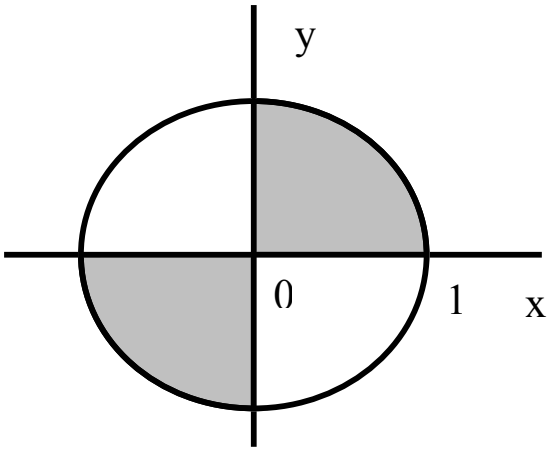
Сложное условие считается выполненным, только если оба условия: 1 и 2 – выполняются одновременно.

3. Логическое отрицание записывается служебным словом *not* ("не"), стоящим перед условием. Оно меняет значение условия на противоположное.

Еще одна логическая операция (*xor,* или "исключающее или") здесь не рассматривается.

Сложные условия, создаваемые с помощью описанных операций, образуют логические выражения. При вычислении их логических значений действуют правила определения приоритета операций, аналогичные арифметическим вычислениям: умножение имеет приоритет перед сложением. Допустимо использование скобок для изменения порядка выполнения действий, причем, как в арифметических вычислениях, общий логический "множитель" можно выносить за скобки. Рекомендуется также при записи логических выражений заключать в скобки отдельные простые условия.

Пример: даны координаты точки на плоскости X и Y. Записать условие, определяющее, попала ли точка в одну из двух заштрихованных областей:



Вариант 1:

(X*X + Y*Y <=1) and (X>=0) and (Y>=0)
or
(X*X + Y*Y <=1) and (X<=0) and (Y<=0)

Вариант 2:

```
(X*X + Y*Y <=1) and
(  (X>=0) and (Y>=0)       or       (X<=0) and (Y<=0))
```

Второй вариант записывается короче.

Более сложные разветвления вычислительного процесса организуются с помощью *оператора выбора*. Его форма:

```
case <выражение> of
        <значение 1> : <оператор 1>;
        < значение 2> : <оператор 2>;
          . . . . .
        < значение N-1> : <оператор N-1>
      else
        <оператор N>;
```

Выражение в заголовке оператора, по которому делается выбор, должно быть целым или символьным. Значения, соответствующие отдельным операторам, по своему типу должны совпадать с типом этого выражения. При исполнении оператора сначала вычисляется значение выражения. Если оно совпадает с одним из перечисленных значений, то выполняется соответствующий этому значению оператор. Если значение выражения не равно ни одному из перечисленных, то выполняется оператор N.

2.3.10. Составной оператор (блок)

Часто бывает необходимо при выполнении какого-либо условия исполнить не один, а несколько отдельных операторов . В этом случае простые операторы объединяются в *составной оператор* или *блок*. В начале блока ставится служебное слово *begin ("начало")*, а в конце слово *end ("конец")* и точка с запятой. Раздел операторов программы также представляет собой блок, в конце которого ставится не точка с запятой, а точка.

Слова *begin* и *end* называются *операторными скобками.* Как и при использовании обычных скобок в арифметических выражениях, количество слов *begin* в программе должно строго соответствовать количеству слов *end*.

2.3.11. Операторы цикла

Значительное количество процессов обработки информации требуют многократного повторения однотипных процессов. Для выполнения таких действий применяются *операторы цикла*. Общая схема работы оператора цикла:

1. Оператор, или последовательность операторов (называемые *телом цикла*), повторяется многократно;
2. При выполнении определенного условия повторение тела цикла прекращается и управление передается следующему по порядку оператору.

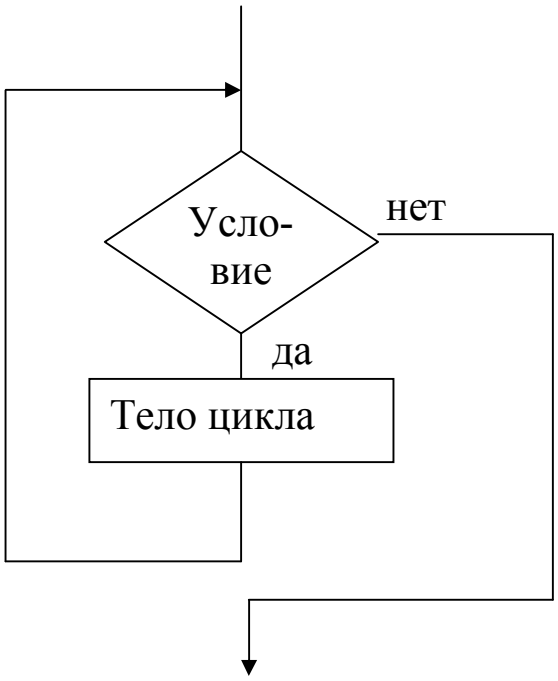
При программировании циклических процессов важно определить как содержание тела цикла, так и условие выхода из него. В зависимости от формирования выхода из цикла в Паскале применяются 3 типа циклов.

А) **Цикл с предусловием** (проверка условия выполнения цикла перед входом в тело цикла):

While <условие> do <тело цикла>;

("Пока") ("выполнять")

Блок-схема такого цикла имеет вид:



Видно, что условие проверяется перед входом в цикл и тело цикла выполняется, только если условие удовлетворяется. Если перед началом цикла условие не выполняется, то тело цикла не будет исполнено ни разу.

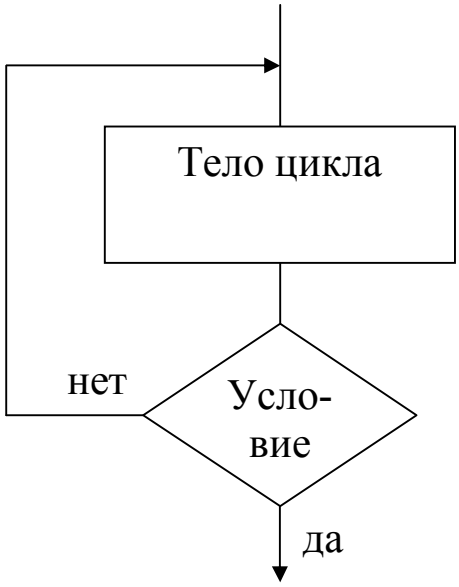
Если тело такого цикла состоит из нескольких операторов, их необходимо объединить в блок.

Важнейшим требованием при использовании описанного цикла является необходимость изменения условия выполнения цикла при исполнении тела цикла. В противном случае цикл будет повторяться до бесконечности (такая ситуация на жаргоне программистов называется "зацикливанием").

Б) **Цикл с постусловием** (проверка условия прекращения повторений в конце тела цикла)**.**

repeat <тело цикла> **until** <условие>;
("повторять") ("до тех пор, пока")

Блок-схема:



Тело такого цикла всегда выполняется хотя бы один раз, а условие проверяется после выполнения тела цикла и при его невыполнении цикл повторяется еще раз. Как и в предыдущем типе цикла, тело цикла должно влиять на условие выхода из него.

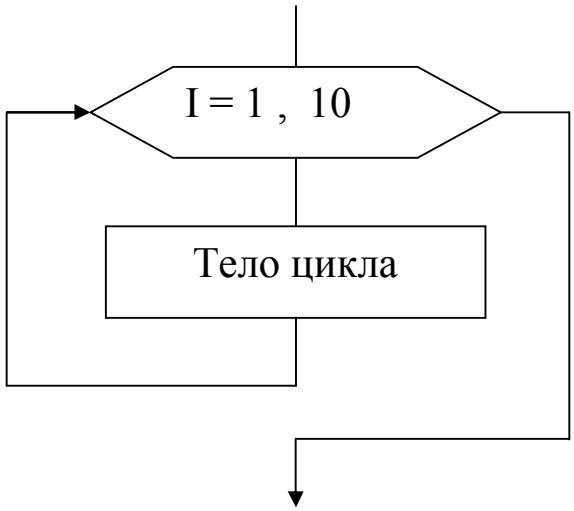
Тело такого цикла может состоять из нескольких операторов, но объединять их в блок не надо, так как служебные слова *repeat* и *until* играют роль операторных скобок.

В) **Цикл со счетчиком.** В отличие от двух вышеописанных, цикл со счетчиком выполняется строго определенное число раз. Его вид:

for <счетчик> := <начало > to <конец > do <тело цикла>;
("для") *("к")* *("выполнить")*

Счетчик – это переменная целого типа, "начало" и "конец" – начальное и конечное значения счетчика (типа *integer* ; конечное значение должно быть больше начального). Вначале счетчику присваивается его начальное значение и выполняется тело цикла, затем счетчик увеличивается на 1 и тело цикла повторяется. Далее значение счетчика увеличивается на 1 после каждого повторения тела цикла, пока после очередного повторения счетчик не станет больше заданного конечного значения. Последний раз тело цикла выполняется при значении счетчика, равном конечному.

Оператор цикла со счетчиком изображается на блок-схемах особым знаком



Переменная I является счетчиком, 1 и 10 – начальное и конечное значения счетчика.

Тело цикла может быть любым оператором или блоком. Важное требование – выполнение тела цикла не должно влиять на значение счетчика.

Разновидностью цикла со счетчиком является цикл вида

for <счетчик>:=<начало> downto <конец> do <тело цикла>;
("вниз к")

Он отличается тем, что начальное значение счетчика должно быть больше конечного и после выполнения тела цикла счетчик каждый раз уменьшается на 1.

Любой цикл можно организовать с помощью условного оператора и оператора перехода (например, для цикла с предусловием: после выполнения тела цикла производится возврат - переход назад к условному оператору, а при нарушении условия – переход к оператору, следующему за телом цикла). Однако для упрощения программирования и придания более ясного вида тексту программ были введены специальные операторы цикла.

2.3.12. Одномерные массивы и операции их обработки

Одномерный массив представляет собой набор однотипных простых значений, объединенных общим именем и различаемых внутри набора по единственному номеру (*индексу*). Математическим аналогом одномерного массива служит последовательность чисел.

Отдельный элемент массива записывается так: ставится общее имя массива и за ним в квадратных скобках – индекс данного элемента: A[1], POSL[N] и т.п.

Массив должен быть описан в разделе описаний программы. Это можно сделать двумя способами

1. Непосредственно описать массив, как переменную.
2. Сначала описать тип данных, имеющих структуру массива, а потом описать сам массив, как переменную описанного типа.

Примеры описаний даны в п. 2.3.5.

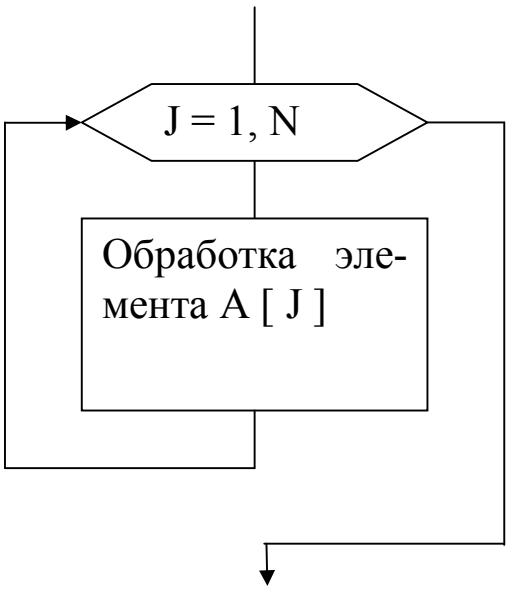
При описании массива количество его элементов должно быть указано как конкретное число – нельзя при описании массива указать число его элементов, как переменную, а потом в программе задать это число, например, вводом с клавиатуры (такие действия были допустимы во многих других языках программирования). Это создает затруднения, если впоследствии предполагается использовать программу для обработки массивов с разным числом элементов. Есть два способа обойти это затруднение:

1. Заранее описать массив с избыточным количеством элементов, а с клавиатуры задавать конкретное число элементов, которые будут практически использованы. Однако при этом приходится использовать лишние ячейки памяти, что не рекомендуется;

2. Описать число элементов, как константу с конкретным значением, а при описании массива указать эту константу в качестве числа элементов массива. Впоследствии при необходимости обработать массив с другим числом элементов надо изменить только значение константы и не потребуется вносить изменения в остальной текст программы. Пример такого описания:

```
. . . . .
const N = 10;
var A: array [1..N] of real;
. . . . .
```

Основным структурным элементом программы, обрабатывающей одномерные массивы, является цикл со счетчиком (так как число исполнений цикла равно числу элементов массива, то есть известно). Если N – число элементов, то отдельный этап обработки массива выглядит так:



Пример: ввод с клавиатуры 5 элементов одномерного массива A [J].

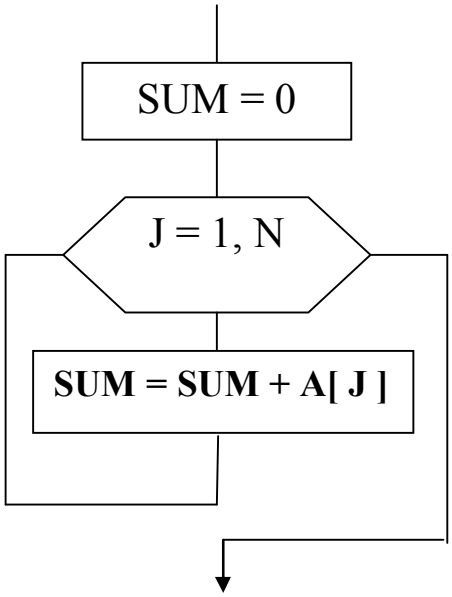
```
. . . . .
for J := 1 to 5 do
   begin
      write ('Введите значение элемента A [ ' ,  J : 3,  ' ] :  ');
         {Пояснительная инструкция для пользователя}
      readln ( A [ J ] );   {Ввод значения с клавиатуры}
```

```
   end;
. . . . .
```

Далее описываются некоторые простые алгоритмы обработки одномерных массивов (во всех примерах N – число элементов массива A).

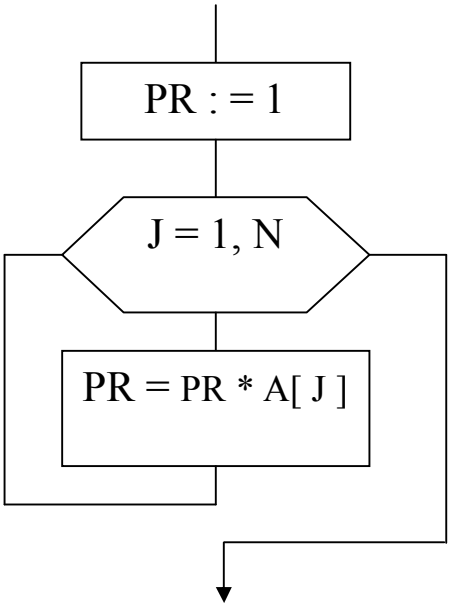
1. Вычисление суммы всех элементов массива (переменная SUM):

```
. . . . .
SUM : = 0;
For J : = 1 to N do
   SUM : = SUM + A [ J ] ;
. . . . .
```



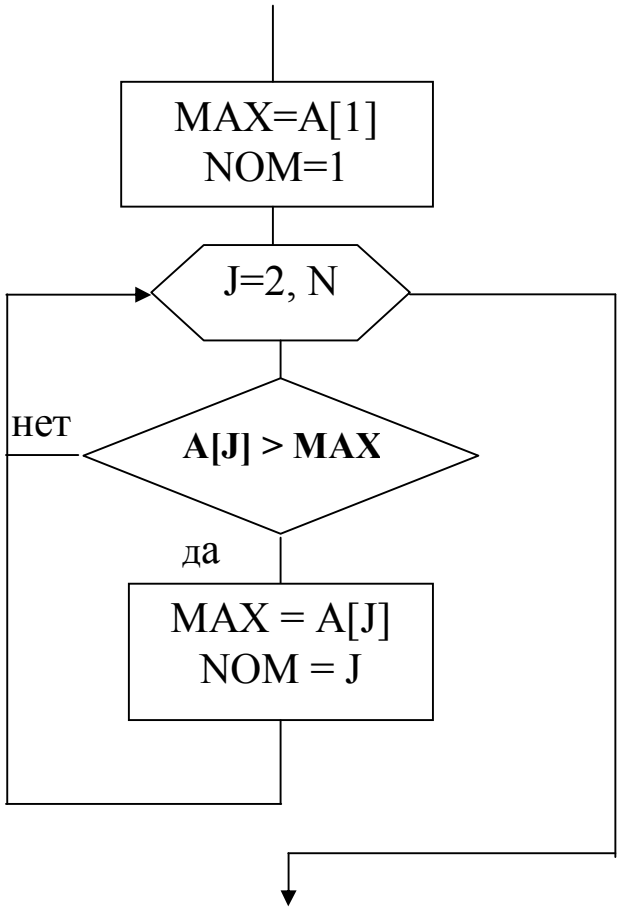
2. Вычисление произведения всех элементов (переменная PR):

```
. . . . .
PR : = 1;
for J : = 1 to N do
   PR : = PR * A [ J ] ;
```



3. Определение максимального элемента массива и его номера (переменные MAX и NOM)

```
  . . . . .
MAX : = A [1] ;
NOM : = 1 ;
for J := 2 to N do
  if  A[J] > MAX then
    begin
      MAX := A[J] ;
      NOM := J ;
    end;
  . . . . .
```



4. Сортировка массива методом "пузырька".

Сортировка – это перестановка элементов массива в порядке возрастания или убывания. Метод "пузырька" – эффективный и простой метод сортировки не слишком больших массивов. Принцип его заключается в следующем (на примере сортировки по возрастанию). Просматривают попарно элементы массива: первый и второй, второй и третий и т.д., и каждый раз, когда последующий элемент оказывается больше предыдущего, их меняют местами. Нетрудно понять, что при этом большой по значению элемент будет постепенно перемещаться в направлении конца массива, пока не встретит на пути элемент, еще больший, затем передвигаться будет этот больший элемент и т.д. После проверки всех пар, вплоть до пары элементов с номерами (N-1) и N, самый большой элемент массива встанет на последнее место. Затем подобным образом проверяют уже первые (N-1) элементов, затем (N-2) и так далее, пока не останется только одна пара элементов: 1-й и 2-й. Переставив их в нужном порядке, получают массив, в котором все элементы выстроены в порядке возрастания.

Алгоритм осуществляется с помощью вложенных циклов. Внутренний цикл – это цикл проверки некоторого количества (M) пар

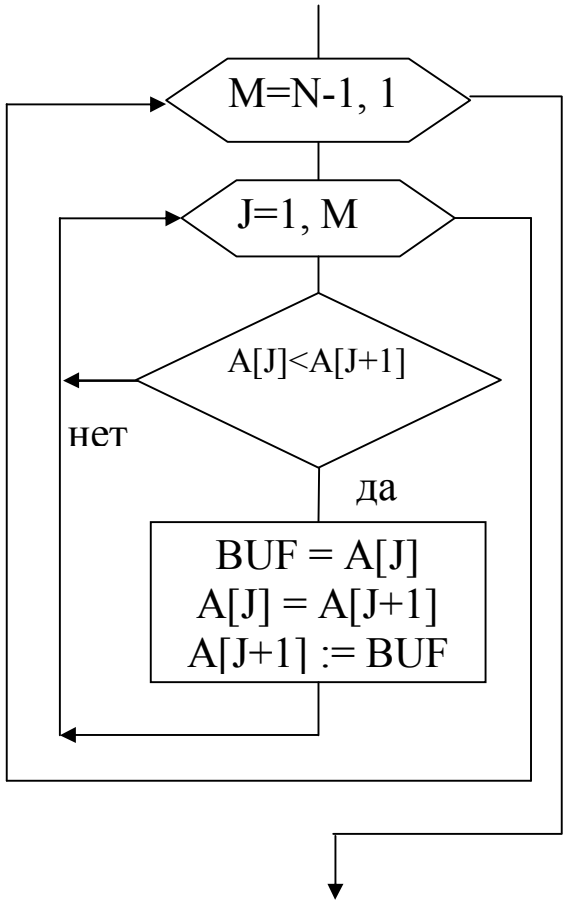
элементов с перестановкой их при необходимости. Внешний цикл позволяет постепенно уменьшать значение M от (N-1) до 1.

Для того, чтобы поменять местами значения элементов в двух ячейках памяти, приходится использовать третью ячейку для временного хранения одного из значений. Эта ячейка (которой в программе соответствует некая переменная) играет роль буфера обмена данными, аналогично буферу ОС Windows. Обмен совершается в три действия:

1) одно из значений переменных записывается в буфер;
2) второе значение записывается в ячейку, соответствующую первой переменной;
3) значение первой переменной из буфера записывается в ячейку, соответствующую второй переменной.

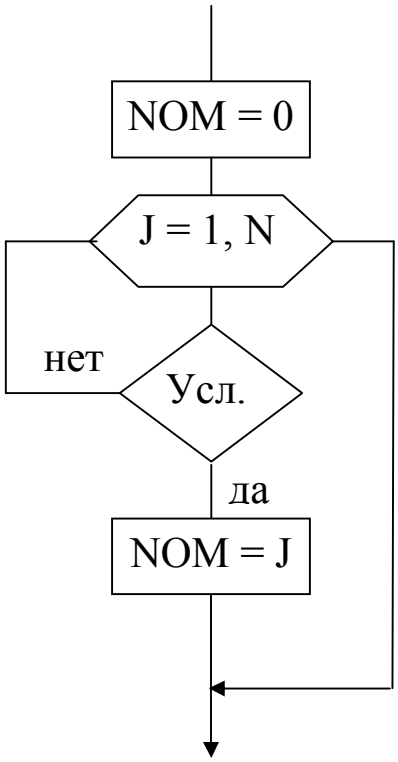
Приведем текст фрагмента программы, описывающего сортировку массива A из 10 элементов по возрастанию. Использованы переменные: J - счетчик внутреннего цикла, M – счетчик внешнего цикла, BUF – переменная-буфер:

```
for M := N-1 downto 1 do
  for J := 1 to M do
    if A[J] < A[J+1] then
      begin
        BUF := A[J] ;
        A[J] := A[J+1] ;
        A[J+1] := BUF ;
      end;
```



5. Поиск первого по порядку элемента, удовлетворяющего какому-то условию (этот и последующие алгоритмы связаны с обработкой элементов, отбираемых по условию. Примеры типичных условий, возникающих при обработке массивов, приведены ниже.)

Для запоминания номера нужного элемента вводят дополнительную переменную и вначале присваивают ей значение 0. Затем просматривают элементы массива, начиная с первого и, встретив элемент, удовлетворяющий условию, присваивают значение индекса элемента упомянутой переменной. Сразу после этого надо прекратить выполнение цикла, что можно сделать с помощью оператора перехода на метку, помечающую следующий за циклом оператор программы (в нижеследующем тексте фрагмента программы 1 – метка)




```
. . . . .
NOM := 0 ;
For J := 1 to N do
  If <условие> then
    Begin
      NOM := J;
      Goto 1
    End;
1:  . . . . .
```

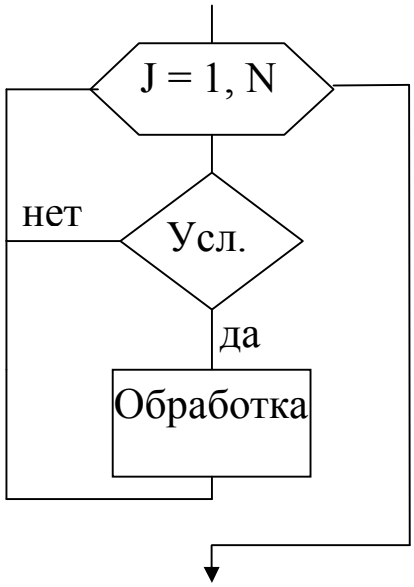
Оператором, помеченным меткой 1, может быть вывод на экран значения NOM. Если оно равно 0, то элементов, удовлетворяющих условию, в массиве нет.

Для решения этой задачи более эффективным является использование цикла с постусловием (т.к. число повторений заранее не известно):

```
. . .
NOM := 0 ;
J:=0;
repeat
J:=J+1;
if <условие> then
  NOM:=J;
```

until <условие>;

. . .



6. Выборочная обработка тех элементов массива, которые удовлетворяют условию, проводится по общему принципу, изображенному участком блок-схемы. От обычных алгоритмов, например, суммирования, эта схема отличается наличием проверки условия перед обработкой очередного элемента. Если элемент этому условию не удовлетворяет, он не обрабатывается, а пропускается. При поиске максимального или минимального элемента, удовлетворяющего условию, предварительным действием служит поиск первого в массиве элемента, удовлетворяющего этому условию. Его значение принимается за предполагаемый искомый максимум (минимум). Если элементов, удовлетворяющих условию, в массиве нет, проводить поиск не имеет смысла.

Пример: подсчитать количество элементов массива из 10 элементов, принадлежащих интервалу (3; 8).

```
program PRIMER;
uses CRT;
const N = 10;
var J, KOL: integer;
     A: array [1 .. N] of real;
begin
  clrscr; {очистка экрана}
       {Ввод}
  for J := 1 to N do
    begin
      write ('Введите '   ,  J:3 ,   '-й  элемент массива : ');
      readln( A[J] );
    end;  {конец этапа ввода начальных значений}
       {Обработка}
  KOL := 0;
  for J := 1 to N do
    if ( A[J] > 3 ) and (A[J] < 8) then
```

```
        KOL := KOL +1;
        {Вывод результата:}
   if KOL = 0 then
        writeln ('Таких элементов в массиве нет' );
      else
        writeln ('Количество элементов равно '    ,  KOL:4);
readln; {задержка окончания программы}
End.
```

7. Определение наличия элемента или группы элементов, удовлетворяющих условию (без подсчета их количества).

Пример: определить, имеется ли в массиве хотя бы одна пара соседних чисел, из которых второе вдвое больше первого.

Решение таких задач можно провести аналогично п. 5, но часто для этого используют "флажок" – переменную логического типа, фиксирующую ответ на вопрос "Имеется ли в массиве нужная группа элементов". Перед началом просмотра массива переменной присваивают значение *false ("нет")*. Затем просматривают массив и, найдя нужную группу элементов, меняют значение "флажка" на *true ("да")*.

```
 . . . . .
var FLAG : boolean; {описание логической переменной
                          в разделе описаний}
 . . . . .
FLAG := false;
for J := 1 to N-1 do
   if ( A [ J + 1 ] = 2 * A [ J ] ) then
      FLAG := true;
if  FLAG then
   writeln (' в массиве есть нужная пара элементов')
      else
         writeln ('нужных пар элементов в массиве нет');
```

Если по ходу просмотра массива встретятся еще группы элементов, удовлетворяющие условию, значение "флажка" уже не изменится, ответ останется неизменным ("да"); если нужная группа не встретится ни разу, сохранится первоначально заданное значение ответа ("нет").

Последний оператор (условный) иллюстрирует использование значения логической переменной в качестве условия.

Примеры типичных условий, накладываемых на элементы массива (A[1 .. N] – массив, J – индекс элемента массива):

а) Отбор элементов, кратных заданному числу K (только для элементов целого типа!):

A [J] mod K = 0

б) Отбор элементов с индексами, кратными K:

J mod K = 0

в) Отбор элементов, значения которых попадают в *интервал* между числами B и C:

(A [J] > B) and (A [J] < C)

г) То же для *отрезка* между числами B и C

(A [J] >= B) and (A [J] <= C)

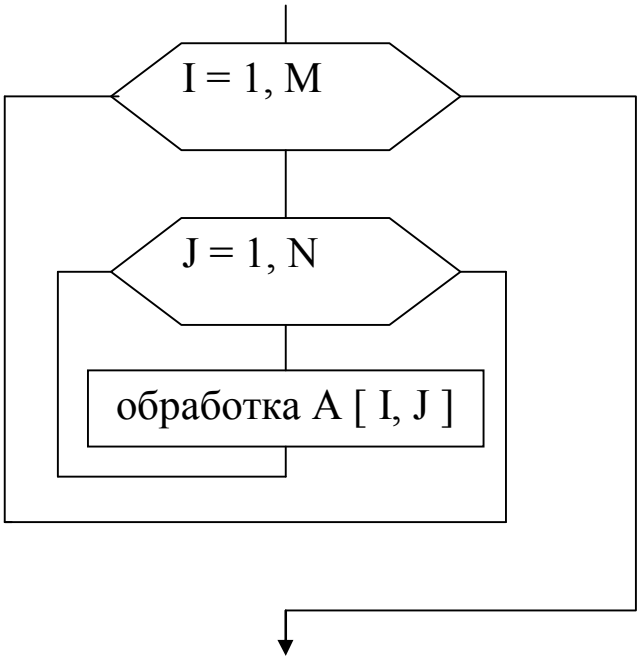
2.3.13. Двумерные массивы и их обработка

Аналогом двумерного массива служит матрица

$$\begin{pmatrix} \mathbf{A_{11}} & \mathbf{A_{12}} & \mathbf{A_{13}} \\ \mathbf{A_{21}} & \mathbf{A_{22}} & \mathbf{A_{23}} \end{pmatrix}$$

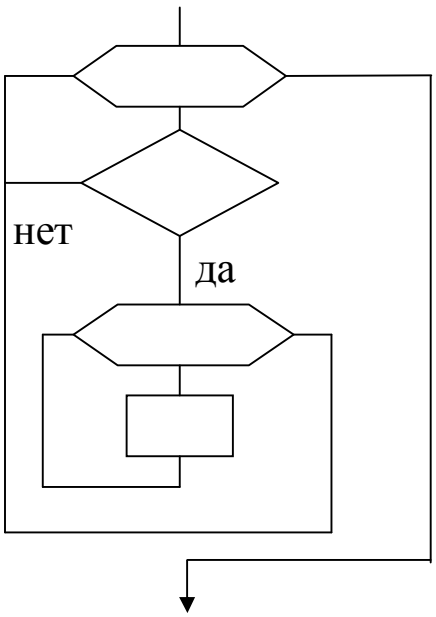
Каждый элемент массива имеет два индекса, из которых первый (традиционно он обозначается **i,** хотя это не обязательно) указывает номер строки матрицы, второй (традиционно – **j**) – номер столбца. Варианты описания двумерного массива приведены в п. 2.3.5. Для обозначения элемента указывают имя массива и за ним в квадратных скобках – индексы через запятую.

Обработка двумерных массивов производится в основном при помощи вложенных друг в друга циклов со счетчиком (M – число строк, N – столбцов)



Если внешний цикл совершается с использованием в качестве счетчика переменной I (как на схеме), то обработка массива ведется "по строкам". При этом внутренний цикл представляет собой обычную обработку I-строки как одномерного массива, элементы которого различаются по индексу J, так что можно использовать алгоритмы обработки одномерных массивов. Если внешний цикл идет по переменной J, то обработка массива идет "по столбцам".

Обработка элементов массива, удовлетворяющих условию, идет аналогично обработке одномерных массивов – проверка условия предшествует оператору обработки очередного элемента. Если условие накладывается на строку или столбец в целом, блок-схема алгоритма имеет вид



Помимо уже перечисленных условий и аналогичных им, укажем еще два:

а) элемент лежит на главной диагонали квадратной (M = N) матрицы (вариант – ниже главной диагонали):

I = J (ниже диагонали: I > J)

б) элемент лежит на побочной диагонали (вариант – ниже нее):

I + J = N+1 (ниже: I + J > N+1).

Пример: в матрице из 3 строк и 4 столбцов найти сумму элементов в каждой строке и вывести на экран справа от матрицы

Очевидно, что значений сумм будет столько же, сколько в матриуе строк, поэтому они образуют одномерный массив из 3 элементов.

Текст программы

```
program SUMMA_V_STROKAH;
uses CRT;
const  M=3;
       N=4;
var    I,J  :integer;
       A: array [1..M, 1..N] of real;
       SUM: array [1..3] of real;
```

```
Begin
  clrscr;
  {Начало этапа ввода данных – значений элементов массива:}
  for I:=1 to M do
    for J:=1 to N do
      begin
        write (' введите A[' ,  I:2,    ' , ' ,  J:2 ,  ']:   ');
        readln(A[I,J]);
      end;
  {Начало этапа обработки:}
  for I:=1 to M do
    begin {внутренность этого блока – обычное суммирование эле-
           ментов одномерного массива – строки; ср. стр. 52}
      SUM[I]:=0;
      for J:=1 to N do
        SUM[I]:=SUM[I]+A[I,J];
    end;
  {Начало этапа вывода результатов:}
  for I:=1 to M do
    begin
      for J:=1 to N do
        write(A[I,J]:8:3); {цикл печати строки матрицы; курсор ос-
                            тается все время в одной строке}
      writeln('             ' , SUM[I]:8:3); {печать суммы с от-
                            ступом и перевод строки}
    end;
  readln;
End.
```

2.3.14. Строки.

Строка (*string*) – это последовательность символов. Длина строки оговаривается при ее описании (см. пример описания в п. 2.3.5, с. 36). При этом не все позиции строки могут быть заняты конкретными символами. Количество имеющихся символов, не вклю-

чающее концевых пробелов, определяет функция *length(S)*, где *S* – переменная типа *string*.

Строка *S* обладает некоторыми свойствами одномерного массива, элементами которого являются отдельные символы, которые можно обрабатывать в программе по отдельности, обозначая *S[i]*. Нумерация элементов начинается с 1. У строки имеется нулевой элемент, в котором хранится информация о количестве символов в ней.

Поэтому при работе со строками можно пользоваться алгоритмами обработки одномерных массивов.

Строки можно сравнивать между собой (если они имеют одинаковую длину) – они считаются равными, если совпадают попарно все их элементы.

При обработке строк можно использовать стандартные функции TurboPascal:

Concat(S1, S2) – функция сцепления строк *S1* и *S2*. Результатом служит новая строка, в которой за последним символом строки *S1* следуют по порядку символы строки *S2*.

Copy(S1,M,K) – функция, результатом которой является фрагмент строки *S1* длиной *K* символов, начиная с символа с номером *M*.

Pos(S2,S1) – функция, результатом которой является номер символа, начиная с которого фрагмент *S2* содержится в строке *S1*. Если в строке *S1* нет такого фрагмента, значение функции *Pos* будет равно нулю.

Для преобразования строк применяются стандартные процедуры:

Delete(S1,M,K); -удаляет из строки *S1 K* символов, начиная с символа с номером *M*.

Insert(S2,S1,M); - вставляет строку *S2* в строку *S1*, начиная с позиции *M* – то есть, первый символ строки *S2* будет иметь в преобразованной строке *S1* номер *M*.

Результатом действия вышеописанных процедур является изменение строки *S1*. Вычисление функций не изменяет строку *S1*.

Пример: Дана текстовая строка из нескольких слов. Определить, сколько в этой строке отдельных слов, начинающихся с приставки 'при'.

Принцип построения алгоритма заключается в том, чтобы последовательно отыскивать в строке фрагменты *' при'*, начинающиеся с пробела (так как речь идет именно о приставке, всегда стоящей в начале слова, то ей должен предшествовать пробел; исключением явля-

ется первое слово, перед которым нет пробела). После обнаружения фрагмента значение переменной-счетчика увеличивается на 1 и найденный фрагмент сразу же удаляется из строки, чтобы он не был по ошибке обнаружен и учтен повторно. Так как при этом строка искажается, операции проводятся не с исходной, а со специально вводимой вспомогательной строкой, которая вначале просто копирует исходную (в эту вспомогательную строку искусственно вставляем пробел перед первым словом). Процесс прекращается, когда при очередном поиске фрагмент не обнаруживается в строке.

Текст программы:

```
program Stroka;
uses CRT;
const PRIST = '  при'; {описание константы строкового типа}
var K : integer; {К - целое число-счетчик}
      S1,S2 : string; {исходная и преобразуемая строки}
Begin
  clrscr; {очистка экрана}
  writeln ('Введите исходную строку: ');
  readln (S1); {ввод исходной строки с клавиатуры}
  S2:=copy ( S1,1,length(S1) ); {Копирование исходной строки}
  insert( '   ', S2, 1); {вставка пробела в начало строки S2}
  K:=0; {начальное значение счетчика}
  While ( pos ( PRIST,S2) > 0 ) do {условие выполняется, если в строке есть нужный фрагмент}
    begin
      K:=K+1; {увеличение значения счетчика}
      delete (S2, pos(PRIST,S2), 4); {удаление найденного фрагмента}
    end;
  writeln ( concat( 'В строке  "  ', S1, '  "  '), K : 3, 'нужных слов.');
  End.
```

Некоторые действия этой программы можно выполнить более экономно, но здесь они иллюстрируют применение различных процедур и функций обработки строк. Если ввести с клавиатуры строку

'Ко мне пришел приятель'

то в результате выполнения программы на экране появится сообщение

В строке " Ко мне пришел приятель " 2 нужных слов.

2.3.15. Подпрограммы. Процедуры и функции.

Подпрограммы – это наборы операторов, предназначенные для неоднократного применения в программе для обработки разных объектов. Подпрограммы делятся на процедуры и функции. Процедуры служат для выполнения различных действий, не всегда связанных с какими-то вычислениями, а функции всегда вычисляют какие-то значения. Команда на выполнение подпрограммы называется обращением к ней, или вызовом подпрограммы. Обращение к процедуре всегда является отдельным оператором программы, тогда как обращение к функции производится в ходе вычисления и значение функции служит операндом при вычислении какого-то выражения.

Выше уже упоминались стандартные процедуры и функции TurboPascal. Рассмотрим основные правила работы с *пользовательскими* (написанными самим программистом) подпрограммами.

Каждая подпрограмма должна быть описана в разделе описаний программы. В описании подпрограммы указываются объекты, которые в ней обрабатываются или используются. На стадии описания эти объекты обозначаются условными именами, называемыми *формальными параметрами* подпрограммы. При обращении к подпрограмме формальные параметры заменяются *фактическими* – именами реальных объектов программы. Типы формальных и фактических параметров должны совпадать.

Если в подпрограмме используются вспомогательные переменные, которые нужны только при ее выполнении и не используются в основной программе, они описываются непосредственно в теле подпрограммы и называются ее *локальными параметрами.* Кроме того, в теле подпрограммы можно без дополнительного описания использовать любые величины, описанные в основной программе – тогда они называются *глобальными параметрами* подпрограммы. Однако широкое использование глобальных параметров является нежелательным.

Структура описания подпрограммы-процедуры:

Procedure <имя процедуры> (список формальных параметров);
< раздел описаний локальных параметров>
begin
<раздел операторов>
end;

Список формальных параметров похож на раздел описаний. Он состоит из перечня формальных параметров, разделенных на группы. Группы отделяются друг от друга точкой с запятой. Каждая группа – это перечень имен формальных параметров, перечисленных через запятую, за которым после двоеточия следует указание типа этих параметров. Перед перечнем имен в группе может стоять ключевое слово *const* или *var*. Наличие этих слов указывает на один из трех видов формальных параметров:

А) параметр-константа (перед списком имен в группе стоит слово *const*). При исполнении процедуры значение фактического параметра, соответствующего формальному параметру-константе, не может изменяться;

Б) параметр-значение (перед списком имен в группе не стоит никаких ключевых слов). В этом случае перед исполнением процедуры начальное значение соответствующего фактического параметра заносится во вспомогательную ячейку памяти, а в конце выполнения процедуры это же значение вновь присваивается фактическому параметру. Таким образом, в ходе выполнения процедуры значение фактического параметра может изменяться, но это изменение будет "потеряно" после окончания выполнения процедуры;

В) параметр – переменная (перед списком имен в группе стоит слово *var*). Значение соответствующего фактического параметра в ходе выполнения процедуры может меняться и это изменение сохранится после завершения процедуры.

Для обращения к процедуре в программе записывается ее имя с указанием в скобках списка фактических параметров. При этом в качестве параметров-констант и переменных могут использоваться только имена (для параметров-переменных – только имена описанных в программе переменных, для констант – имена как констант, так и переменных). В качестве параметров-значений могут использоваться не только имена, но и выражения.

Для примера запишем процедуру, осуществляющую обмен значений между двумя переменными. Такое действие использовалось выше при сортировке одномерного массива (см. п. 2.3.12, с. 54)

```
procedure OBMEN (var AF,BF : real);
  var BUF : real; {описание локального параметра – переменной
                   BUF}
  begin
    BUF := AF;
    AF := BF;
    BF := BUF;
  end;
```

Для обмена значений между двумя элементами массива в программе достаточно теперь записать оператор обращения к процедуре

OBMEN (A[I], A[I+1]);

Описание подпрограммы-функции имеет похожую структуру, но с двумя отличиями. Первое – в структуре заголовка. У функции он имеет вид

Function <имя функции> (<список формальных параметров>) : <тип значения функции>;

Второе отличие заключается в том, что последний оператор в разделе операторов функции имеет вид

<имя функции > := <вычисленное значение функции>;

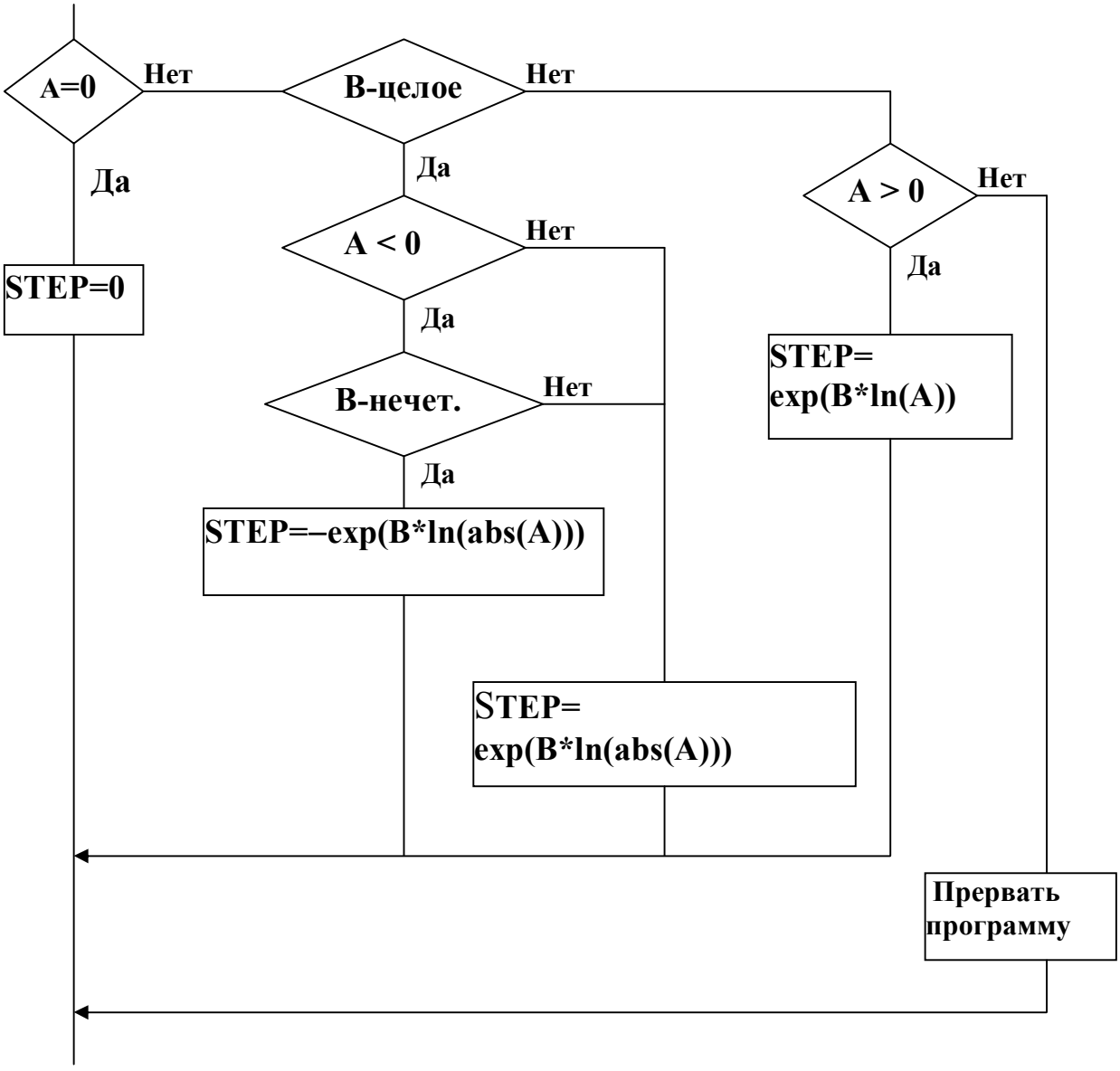
В качестве примера рассмотрим вычисление степенной функции $\mathbf{a^b}$. В языке Pascal отсутствует соответствующая стандартная функция. При вычислении значения степенной функции надо учесть:

1. Отрицательное число можно возводить только в целую степень.
2. Результат возведения в степень с помощью выражения exp(b*ln(a)) всегда является вещественным числом.

Поэтому функцию запишем, указав в ней формальные параметры вещественного типа, значение функции также будет являться ве-

щественным числом. При этом в качестве фактических параметров можно подставлять целые числа любого типа – они автоматически будут преобразованы в вещественные (записаны в форме с плавающей точкой).

Различные варианты вычисления степенной функции можно представить в виде блок-схемы **(STEP** – имя функции)



При попытке возвести отрицательное число в дробную степень происходит прерывание программы.

Текст соответствующей функции:

```
function STEP(A,B: real) : real;
  begin
    if  A = 0  then
      STEP := 0 {Ноль в любой степени – ноль!}
```

```
    else
      if frac(B) = 0 then {Дробная часть B равна нулю, т.е., B - целое}
        begin
          if (round(B) mod 2 = 1) and (A<0) then {B – нечетное, A<0}
            STEP := -exp(B*ln(abs(A)))
          else
            STEP := exp(B*ln(abs(A)))
        end
      else
        if A>0 then
          STEP := exp(B*ln(A))
        else
          begin
            writeln('Вычисление степенной функции невозмож-
                     но');
            readln;
            halt(1);{Прерывание программы}
          end;
end;
```

Дополнительные комментарии к тексту подпрограммы:

1. Функция *round(B)* округляет вещественное число B до ближайшего целого и выдает результат типа *integer*. Т.к. в тексте она применяется к вещественному числу, имеющему целое значение, ее результатом является просто преобразование целого числа B из типа *real* в тип *integer*, для того, чтобы осуществить проверку на нечетность значения B с помощью операции *mod*, применимой только к числам типа *integer*.

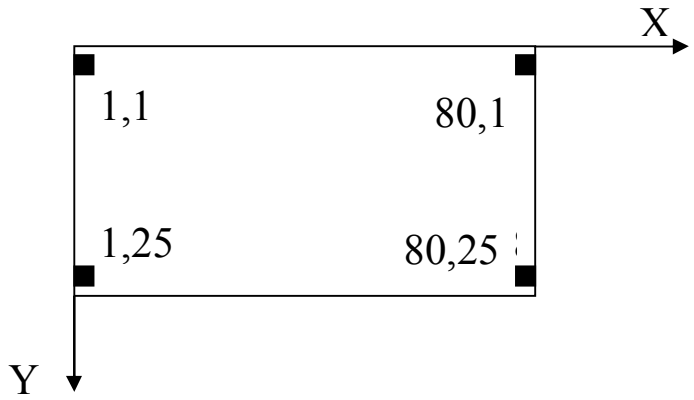
2. Стандартная процедура *halt(1)* принудительно прерывает выполнение программы. Единица в данном случае – так называемый *код прерывания*. Более подробно о прерываниях можно прочитать в детальных описаниях системы TurboPascal.

2.3.16. Модуль CRT. Управление позицией курсора. Окна.

Модуль CRT является одним из стандартных модулей системы TurboPascal. Он содержит подпрограммы, позволяющие управлять вводом и выводом информации при работе монитора в *текстовом* режиме (второй режим – *графический* – здесь не рассматривается).

Подпрограммы модуля CRT в основном можно разделить на две группы – управляющие позиционированием курсора и окон для вывода информации и управляющие цветовыми и звуковыми эффектами. Рассмотрим первую группу подпрограмм.

Для управления установкой курсора для экрана вводится система координат. Горизонтальная координата обозначается X, ее значение может быть целым числом в диапазоне от 1 до 80 и обозначает номер воображаемого столбца на экране (всего экран делится на 80 столбцов). Вертикальная координата обозначается Y и обозначает номер строки (всего на экране может поместиться 25 строк, так что Y – целое число от 1 до 25). Таким образом, координаты отсчитываются от верхнего левого угла экрана и выражаются значениями типа *byte* (см. п. 2.3.1, с. 28)



Для ввода и вывода информации можно создать *окно* – прямоугольную область экрана. Оно создается процедурой

window (XB, YB, XF, YF);

Параметры процедуры: XB, YB – значения координат верхней левой точки окна; XF, YF – координаты нижней правой точки. Эти координаты отсчитываются от левого верхнего угла экрана.

После создания окна все действия по вводу – выводу информации могут совершаться только внутри него; информация, содержащаяся на экране за пределами окна, остается нетронутой. Выйти из окна можно, только открыв новое окно (причем оно может закрыть собой старое). Координатная система также действует только внутри последнего открытого (активного) окна и отсчет координат идет от его верхнего левого угла. Возврат в ранее созданное окно после выхода из него невозможен.

Процедура

GotoXY (XC, YC);

переводит курсор в позицию внутри активного окна, обозначенную координатами XC, YC (важно следить, чтобы значения XC, YC не превышали размеров активного окна). После этого вывод информации на экран будет идти, начиная от позиции установки курсора. Это позволяет грамотно и аккуратно размещать информацию на экране.

При позиционировании курсора можно в качестве координат подставлять встроенные функции TurboPascal

WhereX и *WhereY*

Они не имеют аргументов. Результатом их вычисления являются значения координат X и Y текущего положения курсора. Функциями пользуются для перемещения курсора внутри одной и той же строки или столбца (см. далее пример программы).

2.3.17. Модуль CRT. Управление цветами и звуком.

Для выбора цвета служат процедуры

Textcolor(C);
Textbackground (C);

Первая устанавливает цвет выводимых символов, вторая – цвет фона для этих символов. Параметр C типа *byte* указывает номер цвета согласно списку:

1. Черный (Black)
2. Синий (Blue)
3. Зеленый (Green)
4. Голубой (Cyan)
5. Красный (Red)
6. Фиолетовый (Magenta)
7. Коричневый (Brown)
8. Светло-серый (LightGray)
9. Темно-серый (DarkGray)
10. Светло-синий (LightBlue)

11. Светло-зеленый (LightGreen)
12. Светло-голубой (LightCyan)
13. Светло-красный (LightRed)
14. Светло-фиолетовый (LightMagenta)
15. Желтый (Yellow)
16. Белый (White)

Вместо номера можно подставлять английские названия цветов (приведенные в скобках), так как они закреплены в системе TurboPascal, как стандартные константы.

Процедура

Clrscr;

очищает активное окно от ранее выведенной информации и окрашивает его выбранным цветом фона.

Для управления звуком служат процедуры

Sound (F); и *Nosound;*

Первая из них включает генерацию непрерывного звукового сигнала частотой F Гц. Вторая отключает звуковой сигнал.

Для определения продолжительности сигнала между этими процедурами вставляется еще одна процедура

Delay (T);

Она вызывает задержку исполнения следующего оператора программы (обращения к процедуре *Nosound;*) на T миллисекунд (приблизительно; это зависит от типа компьютера).

Параметры F и T – целые числа, значения которых могут меняться от 0 до 65535 (это дополнительный целочисленный тип *Word*).

Пример: составить таблицу значений функции

$$\mathbf{sh\ x} = \frac{\mathbf{e^{x} - e^{-x}}}{\mathbf{2}}$$ ("гиперболический синус")

на отрезке от 0 до 1 с шагом 0.2. Таблицу вывести на экран внутри расположенного в центре экрана окна белого цвета синими символами. В конце выполнения программы подать звуковой сигнал.

```
program SINGIP;
uses CRT;       {подключение модуля CRT}
const H = 0.2;
var I : Integer;
{Описание подпрограммы-функции:}
function SH ( X: real) : real;
  begin
    SH := ( exp(X) – exp(-X) ) / 2;
  end;   {Конец описания функции}
Begin
  Clrscr; {очистка и заполнение экрана черным цветом, выбираемым
          "по умолчанию"}
  window (34, 8, 48, 17); {создание окна}
  textcolor ( Blue); {выбор цвета символов}
  textbackground ( white ); {выбор цвета фона}
  clrscr; {окрашивание окна выбранным цветом фона}
  {Создание заголовка таблицы:}
  gotoXY (4, 2); {перевод курсора в 4 позицию 2 строки}
  write ('X');
  gotoXY (10, whereY);    {перевод курсора в 10 позицию той же
                           строки}
  write ('sh X');
  {Начало печати таблицы функции}
  for I := 0 to 5 do
    begin
      gotoXY (2, I+4); {курсор переводится при первом исполнении
      цикла в 4-ю строку, при следующем исполнении - в 5-ю и т.д.}
      write ( (I*H) : 4 : 2 );
      gotoXY (10, whereY); {перевод курсора в 10 позицию той же
                            строки}
      write ( SINH(I*H) : 5 : 3 );
    end;
{Генерация звукового сигнала:}
  sound (1000);
  delay (500);
```

```
  nosound;
  readln;
  window (1, 1, 80, 25); {Возврат в режим полного экрана открытием
                          окна размером во весь экран}
  textbackground (black); {установка черного фона для его запоми-
                           нания}
End.
```

Установка черного фона в конце сделана потому, что цветовые установки запоминаются системой TurboPascal безотносительно к конкретной программе и будут действовать по умолчанию на другие программы, что может оказаться нежелательным.

III. ЛАБОРАТОРНЫЕ РАБОТЫ

Курс лабораторных работ включает 4 работы:

1. Программирование линейного вычислительного процесса
2. Программирование процесса обработки одномерного массива
3. Программирование процесса обработки двумерного массива
4. Изучение приемов работы с файлами и папками в ОС Windows.

По работам 1-3 оформляется отчет, включающий:

1. Формулировку цели работы.
2. Математическую постановку задачи
3. Блок-схему алгоритма
4. Текст программы
5. Результаты проверки работы (тестирования) программы.

Цель работы сформулирована в ее описании. Математическая постановка включает перечень используемых в программе переменных и констант с указанием их типов и назначения (исходные данные, промежуточные результаты, конечные результаты), запись формул, по которым ведется расчет.

3.1. Задания для выполнения лабораторной работы 1: Программирование линейного вычислительного процесса

Цель работы: знакомство с общей структурой программы, получение практических навыков написания простейших программ.

1. Период колебаний t маятника длиной L определяется по формуле

$$t = 2\pi\sqrt{\frac{L}{g}}$$

.Вычислить, сколько колебаний совершит маятник за время T.

2. Вычислить силу притяжения **F** между телами массами **m1** и **m2**, находящимися на расстоянии **r** друг от друга:

$$\mathbf{F = G\frac{m1 \cdot m2}{r^2}}\text{, где } \mathbf{G = 6{,}673 \cdot 10^{-11}}\ \text{м}^3/\text{кг} \cdot \text{сек}\ .$$

В качестве одной из масс при тестировании программы использовать массу Земли **m1**=$5.98 \cdot 10^{24}$ кг, радиус Земли **r**=6371 км. При вычислении должна получиться сила тяжести в Ньютонах (Н).

3. Вычислить площадь треугольника, если задан полупериметр *p* и 3 угла (α, β, γ) по формуле $\mathbf{S = p^2 \cdot tg\frac{\alpha}{2} \cdot tg\frac{\beta}{2} \cdot tg\frac{\gamma}{2}}$.

4. Маляру поручили покрасить крышу и стенки бака для бензина в форме цилиндра. Определить, сколько потребуется банок с краской, если известны размеры бака и площадь, которую можно покрасить из одной банки.

5. Изделие имеет форму полого цилиндра, где **T** - толщина стенки, **R** - внешний радиус, **H** - высота, **G** - плотность материала, из которого выполнено изделие. Определить массу изделия и площадь поверхности.

6. Парник имеет форму полуцилиндра, Вычислить количество пленки ($м^2$), необходимой для изготовления парника, если размеры заданы.

7. Из прямоугольного листа вырезают 2 круга, треугольник и прямоугольник заданных размеров. Определить процент используемого материала.

8. Хранилище для сырья имеет вид цилиндра, заканчивающегося полусферой. Определить площадь поверхности, если заданы размеры.

9. Вычислить объем детали и ее массу, если деталь имеет форму прямоугольного параллелепипеда с 4 сквозными отверстиями цилиндрической формы равных размеров. Размеры детали и плотность заданы.

10. Определить, какой процент составляет зеленая зона от площади района, если план района имеет форму прямоугольника, зеленая зона проходит в виде полосы данной ширины вдоль диагонали прямоугольника. Известны длина и ширина прямоугольника и ширина полосы.

11. Три сопротивления соединены параллельно. Найти общее сопротивление соединения.

12. Из квадратного листа картона вырезается коробка в форме куба с известной стороной. Определить процент неиспользованного материала, если размеры листа заданы.

13. Вычислить общую поверхность и объем круглого конуса.

14. Найти периметр правильного n-угольника, описанного около окружности радиусом **r**

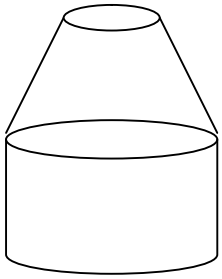
$$\mathbf{p = 2 \cdot n \cdot r \cdot tg\frac{\pi}{n}}$$

15. Вычислить месячную выплату m по займу в S рублей на **n** лет под процент **p** по формуле:

$$\mathbf{m = \frac{r \cdot S \cdot (1+r)^n}{12 \cdot ((1+r)^n - 1)}, \quad где \quad r = \frac{p}{100}}$$

16. Определить, сколько жидкости можно налить в сосуд в форме усеченного конуса.

17. Для хранения и перевозки сырья используются емкости 2-х видов: в форме усеченного конуса и в форме цилиндра. Определить, сколько емкостей 1 и 2 видов потребуется для перевозки 1000 л молока. Размеры емкостей заданы.



18. Сырье хранится в емкости в форме усеченного конуса, совмещенного с цилиндром, определить объем емкости и массу сырья, если размеры емкости и плотность сырья известны. Определить, сколько стального листа (кв. м.) необходимо для изготовления бака в форме цилиндра, если размеры заданы.

19. Для упаковки сырков предложены формочки в виде цилиндра и правильной трехгранной призмы. Размеры (высота, радиус, длина стороны) заданы. Определить количество упаковок 1 и 2 видов, необходимых для расфасовки 100 кг сыра.

20. Определить, сколько стального листа (кв. м.) необходимо для изготовления бака в форме цилиндра, если размеры заданы.

21. Определить массу воздуха, находящегося в цилиндре дизельного двигателя объемом 1,58 л при температуре 67° С и давлении 8· 104 Па. Воспользуйтесь уравнением Менделеева – Клапейрона

$$\mathbf{PV = \frac{m}{k} \cdot R \cdot T}$$

где М = 29 кг/кмоль – молярная масса газа, R = 8314.41 Дж/(кмоль·К) – газовая постоянная, Т – абсолютная температура воздуха, К.

22. В пункте, находящемся на 48 градусе северной широты и 45 градусе восточной долготы, высота Солнца 15 июня определяется формулой:

$$\mathbf{\sin(h) = 0{,}295 + 0{,}614 \cdot \cos(t)}.$$

Здесь $\mathbf{t = 15(T - 1)}$, где **T** – время в часах пополудни. Найти высоту Солнца в момент времени **T** = 8 час (знание высоты Солнца позволяет определить время).

23. «Барометрическая формула», определяющая зависимость атмосферного давления от высоты над уровнем моря, имеет вид:

$$\mathbf{h = 44300 \cdot \left[1 - (p/po)^{0{,}19}\right]},$$

где **h** – высота над уровнем моря в метрах; **p**– давление на данной высоте, мм рт. столба. Вычислить **h** при произвольном давлении (атмосферное давление $\mathbf{p_o}$ = 760 мм рт. столба).

24. За сколько дней бригада численностью **N** человек уберет картофельное поле площадью **S**, если производительность одного человека на уборке картофеля в среднем равна **p**?

25. Из бревна нужно выпилить брус квадратного сечения, имеющий грани шириной по 40 см. Какого диаметра должно быть бревно?

Задания для выполнения лабораторной работы 2: Программирование обработки одномерных массивов.

Цель работы: Знакомство с программированием циклических процессов и алгоритмами обработки одномерных массивов.

1. В массиве чисел X_1, X_2 , ……X_N найти максимальное по абсолютной величине число и его номер.

2. В массиве чисел A_1, A_2 , ……A_N найти минимальное по абсолютной величине число и его номер.

3. В массиве чисел A_1, A_2 , ……A_N найти сумму тех элементов, которые меньше заданного числа D.

4. В массиве чисел B1, B2 , ……BN найти максимальное число и его номер.

5. В массиве чисел C_1, C_2 , ……C_N найти минимальное число и его номер.

6. Для массива чисел X_1, X_2 , ……X_N найти сумму тех элементов, которые больше заданного числа P.

7. . Для массива чисел B_1, B_2 , ……B_N найти произведение элементов, попадающих в заданный отрезок [C, D].

8. Для массива чисел X_1, X_2 , ……X_N найти сумму тех элементов, которые больше первого числа X_1.

9. . Для массива чисел A_1, A_2 , ……A_N найти сумму отрицательных чисел.

10. .Для массива чисел X_1, X_2 , ……X_N найти количество отрицательных чисел.

11. .Для массива чисел X_1, X_2 , ……X_N найти количество чисел, равных заданному числу D.

12. .Для массива чисел A_1, A_2 , ……A_N найти количество положительных чисел.

13. .Для массива чисел A_1, A_2 , ……A_N найти количество нулевых элементов.

14. .В массиве чисел X_1, X_2 , ……X_N отрицательные числа заменить нулями.

15. .В массиве чисел A_1, A_2 , ……A_N изменить знак отрицательных чисел на противоположный.

16. .В массиве чисел $B_1, B_2, \ldots\ldots B_N$ положительные числа заменить нулями.

17. В массиве чисел $C_1, C_2, \ldots\ldots C_N$ найти первое число больше заданного числа.

18. В массиве чисел $B_1, B_2, \ldots\ldots B_N$ найти первое положительное число и напечатать его номер.

19. В массиве чисел $A_1, A_2, \ldots\ldots A_N$ найти первое отрицательное число и напечатать его номер.

20. Для массива чисел $A_1, A_2, \ldots\ldots A_N$ найти сумму положительных чисел среди элементов A_i с чётными номерами.

21. Дан массив $A_1, A_2, \ldots\ldots A_N$. Элементы, стоящие на чётных местах, разделить на 3, а стоящие на нечётных - умножить на 2.

22. Для массива A [1:N] получить массив B [1:N] по правилу :

$$\mathbf{B_i} = \begin{cases} \mathbf{1 \quad , если \ | A_i |> 3} \\ \mathbf{A_i, в \ противном \ случае} \end{cases}$$

23. Для массива A [1:N] все отрицательные числа увеличить на 1, а остальные заменить на 0,7.

24. В массиве X [1:N] найти MIN элемент и его номер среди элементов, кратных 3.

25. Все отрицательные числа последовательности $A_1, A_2, \ldots\ldots A_N$ увеличить на 2,5.

3.3. Задания для выполнения лабораторной работы 3: Программирование обработки двумерных массивов.

Цель работы: Знакомство с программированием вложенных циклов и алгоритмами обработки двумерных массивов.

1. Найти сумму отрицательных элементов в каждой строке матрицы C(5,4) и вывести на экран рядом с матрицей (справа).

2. Определить количество нулевых элементов в каждом столбце матрицы D(5,6) и вывести результат на экран под каждым столбцом.
3. В каждой строке матрицы C(5,5) найти максимальный элемент и заменить его нулем. Полученную матрицу вывести на экран
4. В матрице D(5,5) все элементы на главной диагонали заменить на единицы, вывести на экран полученную матрицу
5. В матрице C(5,5) все нечетные элементы выше главной диагонали увеличить на единицу, а все четные ниже диагонали уменьшить на единицу, полученную матрицу вывести на экран
6. В матрице C(4,5) определить номера строк, в которых количество отрицательных элементов больше двух, вывести на экран вместе с исходной матрицей.
7. В матрице D(6,4) обнулить строки, в которых встречается хотя бы один отрицательный элемент. Вывести обе матрицы на экран.
8. Даны две матрицы A(3,4) и B(3,4). Сформировать матрицу C(3,4) по правилу: $\mathbf{C_{ij} = max(A_{ij}, B_{ij})}$.
9. Определить, в какой строке матрицы A(5,6) больше всего сумма элементов.
10. Выбрать в каждой строке матрицы B(6,5) минимальный элемент и вывести его на экран рядом с матрицей.
11. Найти сумму каждого столбца матрицы A(5,4) и вывести на экран под матрицей.
12. Найти максимальные элементы в каждом столбце матрицы A(4,6) и вывести их на экран под матрицей.
13. Определить номера столбцов матрицы B(4,6), в которых нет ни одного нулевого элемента.
14. В каждом столбце матрицы A(5,4) определить количество нечетных элементов и вывести на экран под матрицей.
15. Все нечетные элементы матрицы B(5,3) увеличить на единицу и вывести на экран обе матрицы.
16. Найти максимальный элемент матрицы A(5,5) и обнулить строку и столбец, на пересечении которых он находится. Обе матрицы вывести на экран.
17. В матрице B(5,6) обнулить элементы, значения которых не попадают в заданный интервал [c,d].
18. Найти в каждом столбце матрицы C(4,6) минимальный элемент, а затем выбрать из них максимальное значение.
19. Найти максимальные элементы в каждой строке матрицы K(5,4), вывести их на экран вместе с их средним арифметическим.

20. Найти номера строк матрицы B(5,7), не содержащих отрицательных элементов и вывести на экран знак (*) рядом с соответствующей строкой матрицы.
21. Найти строку в матрице A(5,6), где больше всего положительных элементов.
22. Найти, в каком столбце матрицы B(N,M) больше всего положительных элементов.
23. Сформировать матрицу B(M,M) по правилу: $\mathbf{B_{i\,j} = i+j}$. Вывести её на экран. На основе этой матрицы сформировать вторую матрицу, сменить знак на противоположный у элементов ниже главной диагонали.
24. Найти максимальный и минимальный элементы в каждой строке матрицы C(7,7) и поменять их местами.
25. Определить номер столбца матрицы G(6,8), в котором меньше всего нулей.

3.4. Содержание лабораторной работы 4: изучение приемов работы с файлами в ОС Windows.

Пользуясь средствами программы Проводник:
1. Создать в папке "Мои документы" папку, именем которой является шифр учебной группы.
2. В этой папке создать вложенную папку, именем которой является фамилия студента
3. Найти на диске **C:** файл, содержащий текст одной из собственных программ на Паскале и скопировать его в папку учебной группы;
4. Скопировать этот файл также в собственную папку;
5. Переименовать файл.
6. Проверить свойства корзины и снять флажок немедленного стирания файлов без заноса в корзину;
7. Удалить файл.
8. Раскрыть корзину, отыскать файл и восстановить его.
9. Удалить папку учебной группы, раскрыть корзину, отыскать в ней папку и окончательно удалить ее.
10. Восстановить первоначальные свойства корзины, если они были изменены в ходе работы (т.е., вернуть флажок немедленного стирания файлов).

IV. УКАЗАНИЯ И ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ КОНТРОЛЬНЫХ РАБОТ.

Ниже приведены задания для выполнения двух контрольных работ (если учебным планом предусмотрена одна работа, в нее включаются задача 1 из первой работы и задача 2 из второй работы). Каждая работа включает по две задачи, для которых приведены по 25 вариантов конкретных заданий. Номер варианта определяется по формуле

$$NM \bmod 25$$

где NM – личный шифр студента (целое число, образованное двумя последними цифрами номера зачетной книжки), то есть по алгоритму:

1. Разделить шифр на 25.
2. Выделить целую часть результата - Р.
3. Номер варианта: № = NM – 25·Р.

Для шифров 00, 25,50,75 номер варианта – 25.

Контрольные работы оформляются в тетради (можно в одной тетради выполнить обе работы). Решение каждой задачи оформляется аналогично отчету по лабораторной работе. Блок-схемы вычерчиваются карандашом. Тексты программ должны быть записаны разборчивым почерком или выполнены чертежным шрифтом. Контрольные работы передаются студентом лично на кафедру или высылаются почтой в деканат заочного факультета КемТИПП не позднее, чем за месяц до начала экзаменационной сессии (по почтовому штемпелю места отправки).

Допускается выполнение контрольных работ в виде текстовых документов с использованием редактора Miicrosoft Word. В этом случае они могут быть отправлены электронной почтой по адресу:

semionov@kemtipp.ru

В качестве темы сообщения (Subject) указывается город, шифр учебной группы и фамилия студента. Сами контрольные работы должны представлять собой файл в формате RTF, содержащий все

необходимые элементы работы – формулировки задач, их постановки, блок-схемы и тексты программ. Срок отправки контрольных работ электронной почтой – не позднее, чем за три недели до начала сессии.

4.1. Контрольная работа 1.

Задача 1 (Циклические процессы). Составить таблицу значений функции **y = f(x)** на отрезке от **a** до **b** с шагом **h.** Вид функции, значения **a, b, h** взять из нижеприведенной таблицы.

Для построения таблицы значений использовать цикл с предусловием для студентов с нечетными значениями личных шифров и цикл с постусловием для студентов с четными значениями личных шифров. Таблица должна выводиться на экран в два столбца с заголовками '**x**' и '**f(x)**'.

Варианты заданий:

	Функция **y = f(x)**	**a**	**b**	**шаг**
1	$x^2 - 3x + 2$	**0**	**4**	**0,25**
2	$x \cdot \sin x$	**0**	**6**	**0,25**
3	$5x^3 e^{-x}$	**0**	**5**	**0,25**
4	$(e^x - e^{-x})/2$	**-2**	**3**	**0.25**
5	$3x^{-2} \cdot \ln x$	**0.5**	**5**	**0.25**
6	$2e^{-x/5} \sin(x/2)$	**0**	**10**	**0.5**
7	$2 \cdot \sin x \cdot \cos x$	**0**	**6**	**0.25**
8	$x^2\sqrt{4 - x^2}$	**-2**	**2**	**0.25**
9	$\sqrt{x} - \ln(x + 1)$	**0**	**4**	**0.25**
10	$\lvert\sin x\rvert$	**-2**	**2**	**0.25**
11	$1/\mathrm{ch}\, x$	**-3**	**3**	**0.25**
12	x^2 / e^x	**0**	**10**	**0.5**
13	$x^4 - x^3 + 2$	**-2**	**2**	**0.25**
14	$e^{\lvert x\rvert} \cdot \cos x$	**-2**	**2**	**0.2**
15	$\sqrt{x^2 + 5x + 3}$	**0**	**4**	**0.25**
16	$e^{-x}\sqrt{x^2 + 5x + 3}$	**0**	**6**	**0.25**

17	$\ln x \cdot e^{-x^2}$	**0.5**	**5**	**0.25**
18	$x + \sin x$	**0**	**5**	**0.25**
19	$x + e^{-x} \cos$	**-1**	**3**	**0.25**
20	$5(\sqrt{x} - \ln x - 1)$	**1**	**3**	**1.1**
21	$5e^{-x^2/2}$	**-5**	**5**	**0.5**
22	$0.5 \sin x \cdot \ln x$	**1**	**5**	**0.25**
23	$\mathrm{sh}\, x \cdot \mathrm{ch}\, x$	**-2**	**2**	**0.25**
24	$\mathrm{th}\, x + 0.25 \sin 2x$	**-2**	**2**	**0.2**
25	$\mathrm{ch}\, x - x^4$	**-2**	**2**	**0.25**

Встречающиеся в заданиях гиперболические функции выражаются:

$$\mathrm{sh}\, x = \frac{e^x - e^{-x}}{2}$$

$$\mathrm{ch}\, x = \frac{e^x + e^{-x}}{2}$$

$$\mathrm{th}\, x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Задача 2 (Разветвленные алгоритмы). Составить программу, выводящую на экран ответ на вопрос, поставленный в соответствии с условиями задачи. Ответ выводится в виде текстовой строки. Если в каком-то варианте необходимо произвести вычисления, результат также выводится на экран.

Варианты заданий:

1. Даны 3 положительных числа. Составьте программу для определения возможности построения треугольника со сторонами, равными указанным числам.

2. Составьте программу для определения принадлежности точки **P** с координатами **(x, y)** заштрихованной области в виде прямоугольника с координатами главной диагонали: (x_1, y_1), (x_2, y_2).

3. Составьте программу для определения принадлежности точки P с координатами **(x, y)** заштрихованной области, которая задана кольцом с центром в начале координат и радиусами R_1 и R_2.

4. Составьте программу для определения принадлежности точки **P** с координатами (**x, y**) заштрихованной области, которая образована первой дугой синусоиды и осью **Ox**.

5. Составьте программу для определения принадлежности точки **P** с координатами (**x, y**) заштрихованной области, которая задана квадратом со стороной **a**, повернутым на 45□, с центром в начале координат.

6. Составьте программу для определения принадлежности точки **P** с координатами (**x, y**) заштрихованной области, которая образована окружностью с центром в точке x_0, y_0 и радиусом **R**.

7. Треугольник задан 3 сторонами. Определить вид этого треугольника (равнобедренный, равносторонний, разносторонний).

8. Выяснить, сколько действительных корней имеет биквадратное уравнение: $ax^4 + bx^2 + c = 0.$

9. Определить, можно ли выпилить из бревна с заданным диаметром брус в форме квадрата с заданной стороной.

10. Определить, поместится ли круг в квадрате, если заданы площадь круга и квадрата.

11. Определить, принадлежит ли треугольник, заданный координатами трех вершин, окружности с центром в начале координат и заданным радиусом.

12. Определить, является ли данное число четным. Если оно четное, определить также, кратно ли оно еще и трем.

13. Определить, имеет ли квадратное уравнение с заданными коэффициентами **a, b, c** действительные корни (то есть, выполняется ли условие: $D = b^2 - 4ac > 0$). Если корни есть – найти их и вывести на экран.

14. Решить систему линейных уравнений по правилу Крамера:

$$a1 \cdot x + b1 \cdot y = c1$$
$$a2 \cdot x + b2 \cdot y = c2$$

Исследовать систему уравнений: сравнить с нулем главный определитель и побочные определители и сделать вывод о наличии решений системы. Если система имеет решение, то его надо найти, если нет – вывести на экран сообщение об этом.

15. Вывести сообщение о том, в какой аквариум войдет больше воды, если его заполнить на ¾. Форма первого – цилиндр с радиусом **R** и высотой **H**, второго – правильная шестигранная призма со стороной **R/2** и высотой **H**.

16. Определить, какое из изделий имеет меньший вес. Изделия имеют форму полого цилиндра с радиусом **R** и высотой **H**. Для первого изделия – полость круглая с радиусом **R/4**, для второго – полость квадратная со стороной **R/2**.

17. Определить, удовлетворяет ли изделие стандарту, если размеры его **x, y, z**. Анализ по допускам выполняется следующими соотношениями: **c<=x<=D; s <= y <= T; z >= P**.

18. Какая бригада за 7 часов выпустит продукции больше? В первой – **N** человек, производительность труда – **K** изделий в час, во второй – **M** человек, производительность – **P** изделий в час.

19. Подоходный налог взимается в зависимости от годового дохода **S**:

$$\mathbf{p} = \begin{cases} \mathbf{0 \quad при \quad S < 20\,000} \\ \mathbf{0.13\,(S - 20\,000) \quad при \quad S > 20\,000} \end{cases}$$

Определить сумму подоходного налога для произвольного дохода **S**.

20. Можно ли из листа с размерами **a** и **b** вырезать круг радиусом **R**? Если можно, то подсчитать процент использованного материала.

21. Маляру поручили покрасить крышу и стенки бака для бензина в форме цилиндра. Определить, достаточно ли будет **N** трехлитровых банок краски или нет, если одной банки хватает на окраску 6 квадратных метров поверхности бака. Если краски не хватает, найти недостающее количество банок. Размеры бака задаются с клавиатуры.

22. Сколько человек в студенческой группе, если **2/3** их количества (что составило **N** человек) присутствовали на собрании? Вывести результат в виде текста с правильно заданным окончанием слова «студенты», например: «В группе **18** студентов». Значение **N** может составлять **12 ÷18**

23. По заданным координатам точки определить, какому квадранту координатной плоскости она принадлежит.

24. Определить, можно ли из листа с размерами **a** и **b** вырезать круг радиусом **R;** если можно, то сколько кругов можно вырезать?

25. Для хранения и перевозки сырья используются емкости 2-х видов: в форме усеченного конуса и в форме цилиндра. Определить, емкостей какого вида потребуется меньше для перевозки 900 литров молока. Размеры емкостей задаются с клавиатуры.

4.2. Контрольная работа 2.

Задачи этой работы имеют смысловое содержание и связаны с обработкой одномерных и двумерных массивов. Вывод результатов на экран должен проводиться для одномерных массивов в виде столбца (столбцов), для двумерных – в виде матрицы. При необходимости на экран должны выводиться также текстовые пояснения. Для оформления вывода использовать подпрограммы модуля CRT.

Задача 1. Обработка одномерных массивов**.**

1 – 6 (Исходные данные)**.** В результате изучения наличия товаров в магазинах города получена информация о видах товаров, имеющихся в магазинах, количестве имеющихся наименований продукции данного вида, количестве наименований импортной продукции данного вида (названия видов, численные данные вводятся с клавиатуры – не менее 5 видов).

Товар	Количество видов	Из них импортных
Колбаса	25	8
.....	.....	.....

1. Напечатать результаты обследования в виде таблицы, заменив информацию в последнем столбце на «% импортных видов» (процентное соотношение должно быть вычислено).

2. Напечатать таблицу только тех товаров, у которых доля импортных видов составляет более 50%.

3. Напечатать таблицу только тех товаров, количество видов которых больше 20.

4. . Напечатать таблицу вида:

Товар	Количество видов	Из них отечественных
Колбаса	25	17
.....	.....	.....

5. Напечатать исходную таблицу и выдать информацию о конкретном товаре по его наименованию, вводимому с клавиату-

ры. Если товара в списке нет, выдать сообщение «товар . . . не обследовался».

6. Напечатать исходную таблицу и определить общее количество обследованных видов и общее количество импортных видов, а также их долю в процентах.

7 – 10 (Исходные данные). По результатам продажи жилья за полгода администрацией области получена следующая информация (всего не менее 5 городов, данные вводятся с клавиатуры):

Город	Кол-во проданных квартир	Общая площадь, кв. м.	Сумма от продажи млн. руб.
Новокузнецк	34	2800	1870
.....	.....	.....	.....

7. Напечатать информацию в виде таблицы, заменив в последнем столбце на «Стоимость 1 кв. м.».

8. Напечатать исходные данные в виде таблицы и составить список городов, в которых количество проданных квартир меньше 20.

9. Напечатать исходные данные в виде таблицы и определить город, в котором стоимость 1 кв. метра минимальна.

10. Напечатать таблицу, заменив последний столбец на «Средняя стоимость 1 квартиры».

11 - 13 (Исходные данные). О работниках фирмы (не менее 5 человек, данные вводятся с клавиатуры) имеется следующая информация: фамилия, количество отработанных часов за неделю, размер почасовой ставки. Во всех вариантах необходимо вычислить их недельную зарплату (количество часов * ставка) и напечатать в виде исходной таблицы:

Ф.И.О.	Проработал часов	Ставка руб.	Зарплата руб.
Иванов	38	25	950
.....	…..	…..	.....

11. Подсчитать общую сумму зарплаты, напечатав исходные данные в виде таблицы.

12. Вывести список работников, количество проработанных часов в неделю у которых меньше 30.

13. Выдать справку о данном работнике (фамилия вводится с клавиатуры). В справке указываются часы и зарплата. Если фамилии работника не оказалось в списке, выдается сообщение об отсутствии данных.

14 – 17(Исходные данные). Компания выпускает несколько видов изделий различной стоимости. Известны названия изделий, объемы заказов по каждому виду изделия и цена единицы каждого изделия. Вывести исходные данные (ввод с клавиатуры, не менее 5 видов) в виде таблицы:

Изделия	Цена одного изделия	Объем заказа (шт.)
Бумага	2000000	10
Тетрадь	400	10000
.....	.....	.....

14. Подсчитать суммарную стоимость всех заказов.

15. Вывести таблицу:

Изделия	Стоимость всего заказа
Бумага	4000
…	…

16. Выдать справку по заданному изделию (название изделия вводится). В справке указываются все сведения и полная стоимость заказа. Если изделие отсутствует, то выдается сообщение об этом.

17. Выдать список только тех изделий с указанием их стоимости, объем заказа которых больше 100 единиц.

18 – 21 (Исходные данные). Фирма располагает следующей информацией: наименование товара, стоимость его единицы, количество единиц (кг, шт., упаковок) в партии, название фирмы-поставщика, поставившего данную партию товара. Данные вводятся с клавиатуры: не менее 5 товаров от не менее, чем 2 фирм-поставщиков (один и тот же товар может поставляться разными поставщиками). На основе исходных данных во всех программах сформировать и вывести таблицу. Кроме того:

18. Напечатать список товаров с их полной стоимостью, поставляемых одной фирмой (название вводится с клавиатуры).

19. Определить самую дешевую партию товара и фирму, его поставляющую.

20. Определить виды товаров, общая стоимость которых не превышает S.

21. Определить, какая фирма (фирмы) поставила товар с наименьшей общей стоимостью.

22 – 25 (Исходные данные). На складе имеются товары различных наименований. Сведения о товарах (ввод с клавиатуры, не менее 5 товаров) вывести в виде таблицы:

Товар	Цена 1 единицы	Кол-во единиц	Вид единицы
мука	150	1000	мешок
Конфеты	15	250	коробка
.....	.....	.....	.....

22. Подсчитать полную стоимость всех хранимых товаров.

23. Вывести таблицу, добавив в нее еще один столбец - «Стоимость товара».

24. Выдать список товаров, хранимых в мешках и сведения о них.

25. Выдать список товаров, единица измерения которых «кг» и цена 1 кг меньше заданного числа S.

Задача 2. (Обработка двумерных массивов).

1 – 7 (Исходные данные). Статистическое управление имеет сведения о стоимости некоторого минимального набора из N продуктов по месяцам года (N>=3). Вывести сведения в виде таблицы с заголовками строк и столбцов (во всех задачах).

Продукт	Стоимость по месяцам											
	1	2	3	4	5	6	7	8	9	10	11	12
Колбаса												
Молоко												
. . . . .												

Кроме того:

1. Найти среднюю стоимость за год каждого продукта и вывести как дополнительный столбец таблицы.

2. Найти стоимость всего минимального набора продуктов в каждом месяце и вывести как дополнительную строку таблицы.

3. .Для каждого продукта найти месяц, в котором стоимость была максимальной, и вывести результаты в виде таблицы: *продукт "..." - максимальная стоимость "..." месяц.*

4. Найти среднюю стоимость всего набора продуктов за год и вывести после таблицы.

5. Найти процентное изменение стоимости каждого продукта за месяц по отношению к предыдущему месяцу, напрмер: ((стоимость за февраль - стоимость за январь)/ стоимость за январь * 100) и вывести в виде таблицы.

6. Найти среднюю стоимость каждого продукта за 1-ое и 2-ое полугодия и вычислить процентное изменение стоимости по полугодиям (см. задачу 205). Результаты вывести после таблицы.

7. Найти процентное изменение стоимости набора продуктов на конец года ((стоимость набора за январь - стоимость набора за декабрь)/ стоимость набора за январь * 100).

8 – 16 (Исходные данные). Информационная таблица по расходу электроэнергии на предприятии по 5 цехам заполняется каждый месяц по мере поступления данных. Вывести таблицу с указанием месяцев и названий цехов.

Цех	Расход энергии по месяцам, МВт-час											
	1	2	3	4	5	6	7	8	9	10	11	12
Литейный	100	…										
Кузнечный	…	…										
. . . . .												

Кроме того:

8. Подсчитать суммарное потребление электроэнергии предприятием за год.

9. Подсчитать суммарное потребление электроэнергии за год каждым цехом и вывести в таблице вместе с исходными данными.

10. Найти среднее потребление электроэнергии за каждый месяц и вывести вместе с исходными данными в таблице.

11. Определить, в каком из цехов было максимальное потребление за год, результат вывести после таблицы исходных данных.

12. Определить, в каком квартале было максимальное потребление электроэнергии предприятием. Вывести таблицу потребления электроэнергии по кварталам и результат задачи.

13. Определить, какой цех и в каком месяце года имел наибольшее потребление электроэнергии. Результат вывести вместе с таблицей исходных данных.

14. Найти среднее потребление электроэнергии каждым цехом в каждом квартале. Результат вывести в виде таблицы.

15. Найти суммарное потребление электроэнергии каждым цехом в 1 и 2-ом полугодиях. Результат вывести в виде таблицы.

16. Определить, в каком квартале и у какого цеха было минимальное потребление электроэнергии.

17 – 23 (Исходные данные). В некоторой компании регистрируется: число часов, отработанных каждым работником в каждый день недели. Известен список работников и почасовая ставка каждого работника. Вывести исходные данные в виде таблицы:

Фамилия	Часов в день						
	1	2	3	4	5	6	7
Иванов	**6**	8	7	6	5	8	0
Петров	…	…	…	…	…	…	…
. . . . .							

Кроме того:

17. Определить суммарное число часов для каждого работника и вывести исходные данные и результаты в виде таблицы.

18. Известна почасовая ставка каждого работника. Определить недельную зарплату каждого работника и вывести результаты вместе с исходными данными.

19. Определить для каждого дня недели суммарное время работы всех сотрудников и результаты вывести вместе с исходными данными.

20. Для каждого дня недели найти число служащих, отсутствующих на работе, и вывести результаты вместе с исходными данными.

21. Вывести список работников, отработавших менее 30 часов в неделю.

22. Определить фамилию работника с наименьшим числом отработанных часов (не считая 0) и вывести результат вместе с таблицей исходных данных.

23. Найти день недели, в который было отработано наименьшее число часов всеми работниками.

24 - 25 (Исходные данные). В таблице приведено время выпечки хлебобулочных изделий на печах различного типа. Данные вводятся с клавиатуры и должны быть выведены на экран в виде таблицы:

Вид Изделия	**Время в часах для разных типов печей**				
	Тип 1	Тип 2	Тип 3	Тип 4	Тип 5
Хлеб бел.					
Хлеб черн.					

. . .					

Кроме того:

24. Определить для каждого типа изделий предпочтительный тип печи (с минимальными затратами времени на выпечку) и вывести результаты вместе с таблицей исходных данных.

25. Подобрать для каждого типа печи предпочтительный тип изделия (с минимальными затратами времени на выпечку) и вывести результаты вместе с таблицей исходных данных.

V. СОДЕРЖАНИЕ ЭКЗАМЕНА

Экзамен состоит из двух частей – теоретического опроса и практического задания. В теоретической части студенты должны ответить на один из нижеперечисленных вопросов. Практическая часть включает создание несложной программы на языке Паскаль. Содержание программы не выходит за рамки материала, описанного в пособии, в том числе рассмотренного в лабораторных работах. В случае приема экзамена без применения компьютера допускается просто запись текста программы на бумаге, при этом мелкие грамматические погрешности в расчет не принимаются. При использовании компьютера программа должна быть выполнена на компьютере с исправлением ошибок (т.е., должна работать).

ВОПРОСЫ ТЕОРЕТИЧЕСКОЙ ЧАСТИ:

1. Информация и ее цифровая запись в памяти компьютера. Двоичная система счисления. Виды информации.

2. Основные части компьютера.

3. Понятия файла, папки, дерева папок. Атрибуты файла.

4. Виды программного обеспечения компьютеров.

5. Общие принципы устройства ОС Windows

6. Работа с меню. Виды меню. Диалоговые окна.

7. Окна в ОС Windows, их виды, основные элементы. Панели инструментов. Способы выполнения команд.

8. Буфер обмена. Команды работы с буфером.

9. Рабочий стол и его элементы. Пиктограммы и ярлыки. Панель задач.

10. Устройство Проводника. Работа с файлами в Проводнике.

11. *FAR manager* и работа с ним.

12. Основные характеристики языка Паскаль. Типы данных, обрабатываемых в Паскале, управляющие структуры, алфавит языка.

13. Структура программы на Паскале. раздел описаний

14. Простые операторы Паскаля. Ввод и вывод данных на Паскале.

15. Условный оператор и его разновидности.

16. Операторы цикла.
17. Одномерные массивы и их обработка.
18. Двумерные массивы и их обработка.
19. Строки и их обработка.
20. Подпрограммы.
21. Модуль CRT.

Семенов Андрей Германович

ИНФОРМАТИКА

Редактор Л.Г. Барашкова
Худож. редактор Л.П.Токарева

Лицензия 020524 от 2.06.97. Подп. в печать . Формат 60×84/16. Отпечатано на ризографе. Уч.-изд. л .
Тираж экз. Заказ № . Цена р

Кемеровский технологический институт пищевой промышленности
650056, г. Кемерово, бульвар Строителей, 47.
Отпечатано в лаборатории множительной техники КемТИПП,
650010, г. Кемерово, Красноармейская, 52.