ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение
высшего профессионального образования
«Томский политехнический университет»

О. Е. Мойзес, А. В. Кравцов

ИНФОРМАТИКА

Часть 1

Учебное пособие

Третье издание, переработанное и дополненное

Издательство ТПУ
Томск 2007

УДК 519.682(075.8)
М 74

Мойзес О. Е., Кравцов А. В.

М 74 Информатика. Часть 1: учебное пособие. – 3-е изд., перераб. и доп. – Томск: Изд-во ТПУ, 2007. – 127 с.

В учебном пособии рассмотрены вопросы архитектуры ЭВМ, особенности работы в системах MS DOS, WINDOWS. Приводятся основы программирования на алгоритмическом языке Паскаль. Основные приемы и методы программирования рассмотрены на конкретных химических задачах. Приведены типовые алгоритмы для решения задач различных классов.

Пособие подготовлено на кафедре химической технологии топлива и химической кибернетики ТПУ и предназначено для студентов ИДО, обучающихся по направлениям 240100 Химическая технология», 241000 «Энерго- и ресурсосберегающие процессы в химической технологии, нефтехимии и биотехнологии».

УДК 519.682(075.8)

Рекомендовано к печати Редакционно-издательским советом Томского политехнического университета

Рецензенты:

Н. В. Юдина – канд. техн. наук, старший научный сотрудник института Химии нефти СО РАН;

В. Л. Каминский – канд. техн. наук, доцент Томского университета автоматизированных систем управления и радиотехники.

1. ЧТО ТАКОЕ ИНФОРМАТИКА

1.1. Информатизация общества

Отыскание рациональных решений в деятельности людей требует обработки больших объемов информации. Это невозможно без привлечения специальных технических средств. Возрастание объема информации особенно стало заметно в середине XX в. Громадный поток информации не дает нам возможности воспринимать эту информацию в полной мере. В результате наступает информационный кризис, который проявляется в противоречии между возможностями человека по восприятию и переработке информации и мощными потоками информации. Информационный кризис поставил общество перед необходимостью поиска путей выхода из создавшегося положения. Внедрение ЭВМ в различные сферы деятельности послужило началом нового эволюционного процесса, называемого *информатизацией* [1, 2].

Информатизация общества **–** организованный социально- экономический и научно-технический процесс создания оптимальных условий для удовлетворения информационных потребностей и реализации прав граждан, органов государвственной власти, организаций и т. д. на основе формирования и использования информационных ресурсов.

1.2. Информатика – предмет и задачи

Термин «*информатика*» возник в 60-х гг. во Франции. Французский термин *informatique* (информатика) образован слиянием слов *information* (информация) и automatique (автоматика) и означает «автоматизированная переработка информации».

Выделение информатики как науки (отдельной области человеческой деятельности) связано в первую очередь с развитием компьютерной техники. Основная заслуга в этом принадлежит микропроцессорной технике, появившейся в середине 70-х гг.

Информатика – это наука, которая изучает общие законы, методы накопления, передачи и обработки информации с помощью ЭВМ [1, 2].

Информатика появилась благодаря компьютерной технике, базируется на ней и немыслима без неё. Источники информатики: документалистика (сформировалась в конце XIX в., предмет – изучение рациональных средств и методов повышения эффективности документооборота) и кибернетика (1948 г., основоположник – Н. Винер).

Под ***информацией*** понимают любые сведения об объективно существующих объектах и процессах, их связях и взаимодействии, доступные для практического использования в деятельности людей [2].

Информатика как *фундаментальная* наука занимается разработкой методологии создания информационного обеспечения процессов управления любыми объектами на базе компьютерных информационных систем [2]. Цель фундаментальных исследований в информатике – получение обобщенных знаний о любых информационных системах, выявление общих закономерностей их построения и функционирования.

Информатика как *прикладная* наука занимается:

– изучением закономерностей в информационных процессах;

– созданием информационных моделей коммуникаций в различных областях человеческой деятельности;

– разработкой информационных систем и технологий в конкретных областях и выработкой рекомендаций относительно их жизненного цикла.

Главная функция информатики заключается в разработке методов и средств преобразования информации и их использовании в организации технологического процесса переработки информации.

Задачи информатики в следующем:

– исследование информационных процессов любой природы;

– разработка информационной техники и создание новой технологии переработки информации;

– решение научных и инженерных проблем создания, внедрения и обеспечения эффективного использования компьютерной техники и технологии во всех сферах общественной жизни.

Информатика является комплексной научно-технической дисциплиной, призванной создавать новые информационные техники и технологии для решения проблем в других областях.

В информатике в узком смысле можно выделить три части [1–3]: технические средства, программные средства, алгоритмы и теоретические методы решения задач на ЭВМ.

1.3. Знакомство с вычислительной машиной

В настоящее время существует огромное разнообразие электронных вычислительных машин (ЭВМ) различных типов и марок. ЭВМ представляет собой устройство (точнее, совокупность взаимосвязанных устройств), которое способно выполнять определенный набор элементарных арифметических и логических операций. Выполнив одну операцию, ЭВМ автоматически переходит к выполнению следующей и, таким

образом, может выполнить длинные цепочки операций без вмешательства человека.

В составе вычислительной машины различают аппаратуру и программное обеспечение. К последнему относят совокупность программ, определяющих функционирование аппаратуры [1–3].

Общий вид персонального компьютера представлен на рис. 1. Функциональная схема вычислительной машины приведена на рис. 2. Назначение основных, а также периферийных устройств компьютера будут далее рассмотрены.

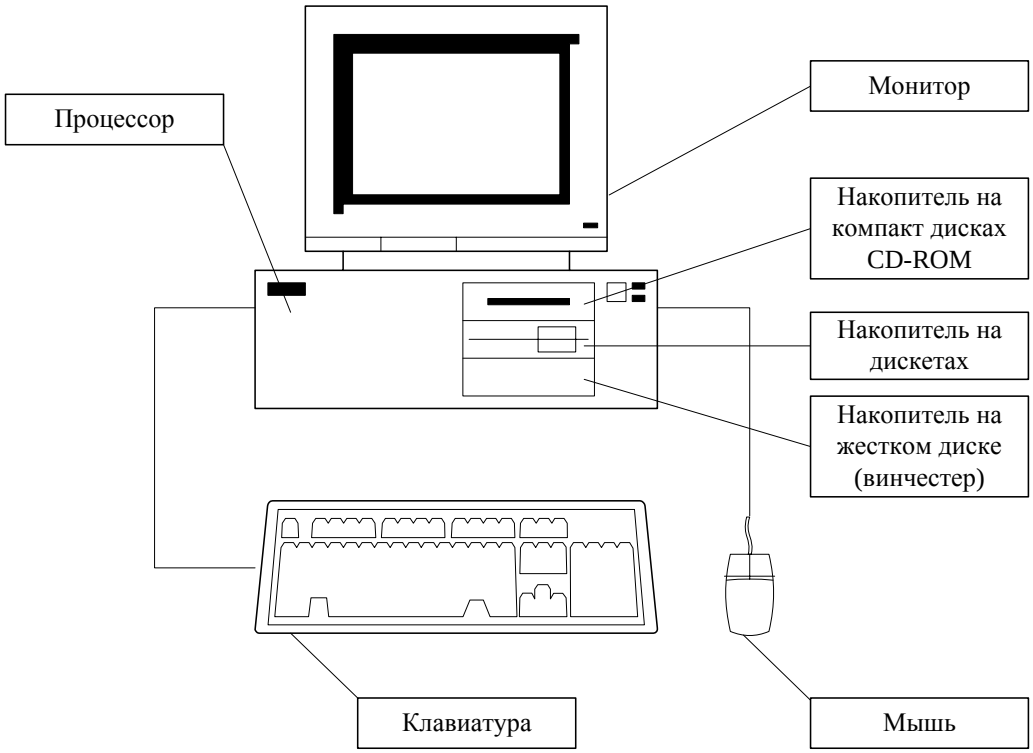


Рис. 1. Общий вид персонального компьютера

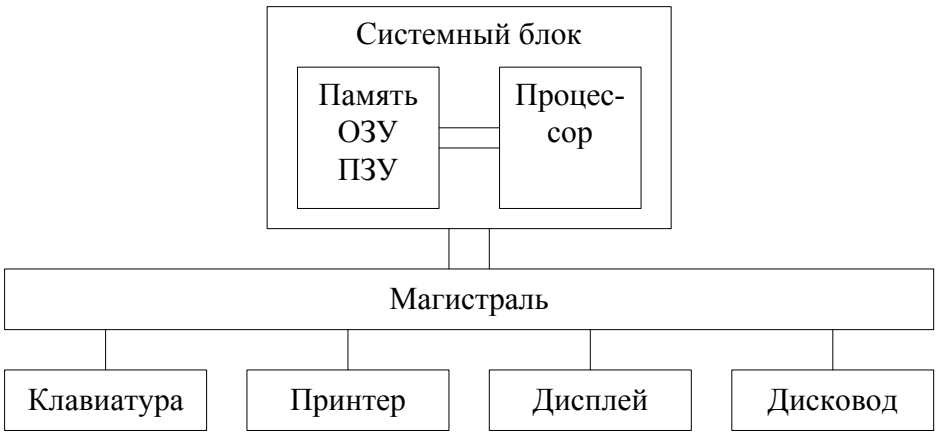


Рис. 2. Функциональная схема ЭВМ

1.4. Представление информации в ЭВМ

ЭВМ обрабатывает информацию только в закодированном виде. Информация называется закодированной, если любая ее элементарная часть представлена в виде числа. Такие числа называются кодами. Вся информация, циркулирующая в центральном устройстве, имеет двоичное представление, т. е. записана в виде последовательности из нулей и единиц, что вызывает необходимость кодирования и декодирования информации. Запись символов в двоичной системе более громоздка, чем в десятичной, но более приспособлена к использованию в компьютерах, в которых имеется два состояния: наличие сигнала (1) и отсутствие сигнала (0).

Пример записи некоторых чисел в двоичном виде:

0 – 0	4 – 100	8 – 1000
1 – 1	5 – 101	9 – 1001
2 – 10	6 – 110	10 – 1010
3 – 11	7 – 111	11 – 1011

Минимальная единица информации – ***бит*** (***bit***). Один бит информации – это одна двоичная цифра: 0 или 1. Это очень маленькое количество информации, поэтому в компьютерах для обработки информации используется более крупная единица – ***байт*** (***byte***), 1 байт = 8 битам.

Байт – это также элементарная ячейка памяти ЭВМ. Каждая ячейка имеет адрес (номер ячейки) и содержимое (двоичный код), которое хранится в ней.

Для измерения памяти ЭВМ используются также килобайты и мегабайты:

1 Кбайт = 1024 байтам;

1 Мбайт = 1024 Кбайтам.

При обработке информации процессор находит по адресу нужную ячейку, читает из нее содержимое, выполняет необходимые действия и записывает результат в другую ячейку памяти.

1.5. Этапы решения задач на ЭВМ

Процедуру подготовки и решения задач на ЭВМ можно представить в виде следующих этапов:

1. Постановка задачи. Задача формулируется пользователем или выдается ему в виде специального задания. На данном этапе осуществляется выбор общего подхода к решению задачи, определение конечной цели.

2. Математическая формулировка задачи, т. е. представление ее в виде уравнений, соотношений и т. д.

3. Выбор метода решения (процедуры), позволяющего свести решаемую задачу к последовательности элементарных действий.

4. Разработка алгоритма решения задачи.

5. Написание программы на языке программирования. На данном этапе происходит перевод разработанного алгоритма на язык программирования.

6. Ввод программы и исходных данных в ЭВМ.

7. Отладка программы. На этом этапе производится обнаружение с помощью ЭВМ ошибок в программе и их исправление.

8. Решение задачи на ЭВМ. Обработка и интерпретация результатов расчета в соответствии с поставленной задачей.

2. ПРОГРАММНО-ТЕХНИЧЕСКИЕ СРЕДСТВА ИНФОРМАТИКИ

2.1. Назначение основных и периферийных устройств компьютера

Персональный компьютер включает следующие основные устройства [4]:

– процессор, выполняющий управление компьютером, вычисления и т. д.;

– клавиатуру, позволяющую вводить символы в компьютер;

– монитор (дисплей) для изображения текстовой и графической информации;

– накопители (дисководы) для гибких магнитных дисков, используемые для чтения и записи на гибкие магнитные диски (дискеты);

– накопитель на жестком магнитном диске, предназначенный для чтения и записи на несъемный магнитный диск (винчестер).

Для передачи информации между системным блоком и внешними устройствами используется магистраль.

Рассмотрим более подробно назначение, структуру и разновидности основных устройств компьютера.

2.1.1. Процессор

Процессор персонального компьютера IBM PC содержит следующие элементы:

– основной микропроцессор, управляющий работой компьютера и выполняющий все вычисления. Наряду с основным микропроцессором большинство современных компьютеров содержат математический сопроцессор, предназначенный для выполнения операций с числами с плавающей запятой. При этом скорость расчетов возрастает в пять и более раз;

– оперативную память, в которой располагаются программы, выполняемые компьютером, и используемые программами данные;

– электронные схемы (контроллеры), управляющие работой различных устройств, входящих в компьютер;

– порты ввода-вывода, через которые процессор обменивается данными с внешними устройствами.

Оперативная память. Оперативная память компьютера разделяется на несколько типов: conventional, upper, extended, high.

Conventional – базовая память. Под базовой памятью понимают первые 640 Кбайт оперативной памяти компьютера. В младшие ее адреса загружается операционная система и драйверы устройств. Этот размер памяти называется lower. Оставшуюся свободную часть занимают пользовательские программы.

Upper. Это область памяти между 640 Кбайт и 1 Мбайт. Часть этой памяти используется для загрузки видеоадаптера. При использовании специальных программ эта часть памяти становится доступной для загрузки пользовательских программ.

Extended. Раздел памяти свыше 1 Мбайт принято называть extended. С помощью специальных программ пользователь может использовать эту память для создания временного логического диска, как буфер для ускорения обмена с жестким диском и т. д.

High. Так называют первые 64 Кбайта extended памяти.

Expended. Для использования памяти свыше 640 Кбайт в прикладных программах фирмы Lotus, Intel, Microsoft разработали стандарт LIM EMS (Lotus, Intel, Microsoft Expanded Memory Specification), позволяющий адресовать до 32 Мбайт оперативной памяти.

Для достаточно быстрых компьютеров необходимо обеспечить быстрый доступ к оперативной памяти. Для этого такие компьютеры могут оснащаться **кэш-памятью**, т. е. «сверхоперативной» памятью относительно небольшого объема, в которой хранятся наиболее часто используемые участки оперативной памяти. Кэш-память располагается «между» микропроцессором и оперативной памятью, и при обращении микропроцессора к памяти сначала производится поиск нужных данных в кэш-памяти.

2.1.2. Магнитные носители и накопители

Основным носителем информации IBM PC-совместимого компьютера является магнитный диск. Различают жесткий магнитный диск (винчестер) и гибкий магнитный диск (дискета или флоппи-диск).

Накопители на жестком диске предназначены для постоянного хранения информации, используемой при работе с компьютером: программ операционной системы, часто используемых пакетов программ, редакторов документов, трансляторов с языков программирования и т. д.

Гибкие диски позволяют переносить программы и документы с одного компьютера на другой, хранить информацию, не используемую постоянно на компьютере, делать архивные копии информации, содержащейся на жестком диске. Могут быть использованы два типа дискет: флоппи-диск 5,25 дюйма (в настоящее время практически не используются) и флоппи-диск 3,5 дюйма (рис. 3, 4). Для чтения/записи гибких дисков служат внешние магнитные накопители, называемые дисководами (рис. 5, 6).

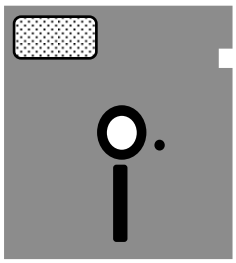


Рис. 3. Дискета размером 5,25 дюйма

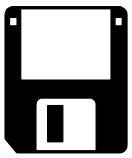


Рис. 4. Дискета размером 3,5 дюйма

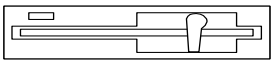


Рис. 5. Дисковод для дискет размером 5,25 дюйма

Рис. 6. Дисковод для дискет размером 3,5 дюйма

Магнитные ленты и накопители на них. В настоящее время используются только для архивного хранения больших объемов информации. Наиболее массовый вид устройств для этой цели – кассетные накопители, так называемые стримеры. Стримеры могут подключаться к контроллеру дисковода для гибких дисков и позволяют сохранять на ленте 250, 510 или 680 Мбайт данных.

Оптические (лазерные) компакт-диски (CD) и дисководы. Наиболее распространены диски «только для чтения» – CD-ROM и дисководы для них. В этих устройствах применяется технология цифровой записи, для чтения и записи данных используют лазерный луч.

Благодаря этому CD обладают достаточно большой скоростью работы и чрезвычайно высокой емкостью: от 127 Мбайт до 600 Мбайт.

Bernoulli. Накопители на гибких магнитных носителях кассетного типа большой емкости (до 230 Мбайт). Их работа основана на аэродинамическом эффекте («эффекте Бернулли») между головкой чтения/записи и магнитной поверхностью. Bernoulli-накопители являются надежными, быстродействующими и достаточно дорогими устройствами.

Магнитооптические диски и дисководы. Основаны на записи данных при помощи дополнительного магнитного поля (поля смещения) и луча лазера. Емкость магнитооптических дисков достигает сотен и тысяч мегабайт.

2.1.3. Видеосистема

Видеосистема считается стандартным устройством вывода IBM PC. Видеосистема состоит из ***дисплея*** и ***видеоконтроллера*** (видеоадаптера) [5]. Дисплей (рис. 7) является достаточно простым устройством на основе электронно-лучевой трубки и во всем подчиняется командам, поступающим с платы видеоадаптера. Изображение, создаваемое видеоадаптером, может быть текстовым и графическим, поэтому различают текстовый и графический режимы работы. В графическом режиме работы на экран выводится любое изображение, состоящее из точек, в текстовом режиме выводятся только символы ASCII-кодировки, но с более высокой скоростью. Текущее место вывода на экран в текстовом режиме всегда отмечается курсором. Видеосистема компьютера характеризуется целым рядом показателей: разрешающей способностью, палитрой, числом видеорежимов и видеопамятью. В зависимости от этих показателей различают следующие видеоадаптеры: MDA (Monochrome Display Adapter), CGA (Color Graphics Adapter), HGC (Hercules Graphics Card), EGA (Enhanced Graphics Adapter), VGA (Video Graphics Array), SuperVGA (SVGA).

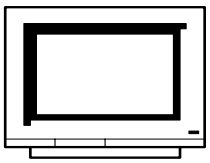


Рис. 7. Дисплей

2.1.4. Клавиатура

Клавиатура IBM PC предназначена для ввода в компьютер информации от пользователя [4]. На рис. 8 показана клавиатура для IBM

PC AT, для других моделей компьютера расположение и число клавиш может несколько отличаться, но назначение одинаковых клавиш на различных компьютерах совпадает. Все клавиши на клавиатуре можно разбить на 4 группы: алфавитно-цифровая часть клавиатуры; функциональные клавиши; клавиши управления курсором; цифровая клавиатура.

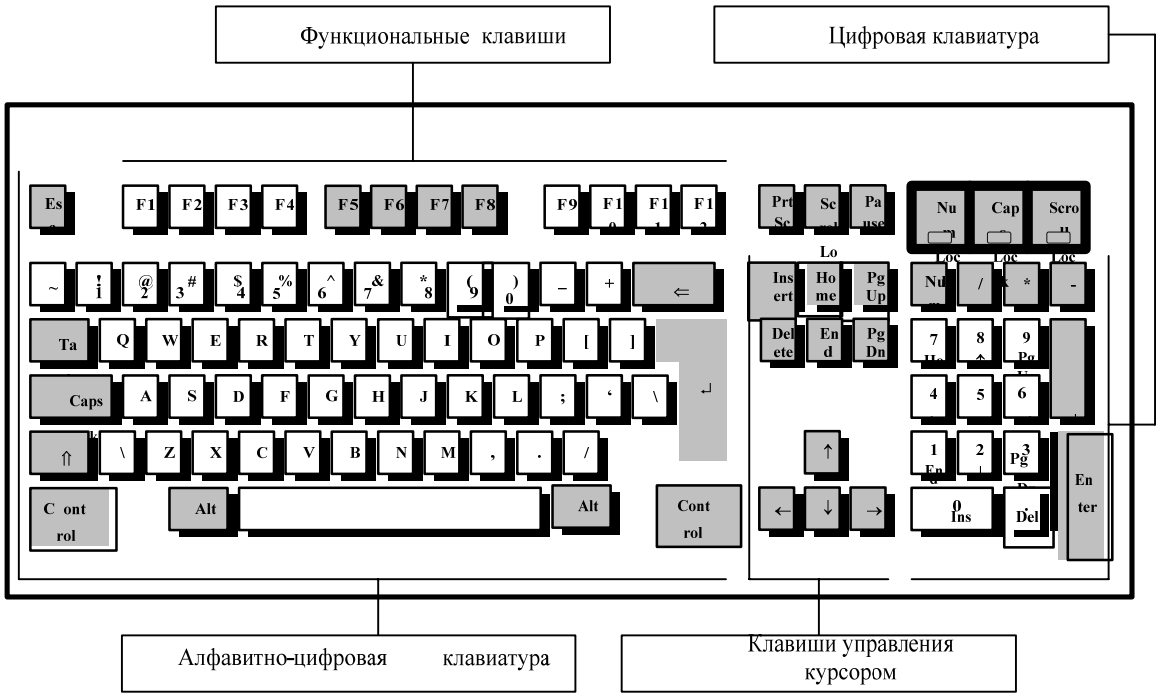
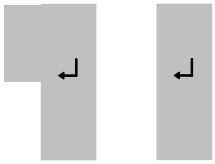
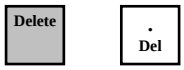
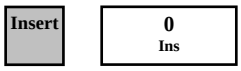
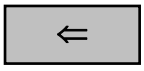
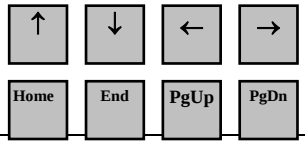
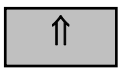


Рис. 8. Клавиатура компьютера IBM PC
(модель клавиатуры для IBM PC AT)

Хотя применение клавиатуры в той или иной программе бывает совершенно индивидуальным, следует рассмотреть некоторые специальные клавиши клавиатуры (см. табл. 1).

Таблица 1

Клавиша	Назначение
1	2
	Клавиша **Enter** (Return или CR) предназначена для окончания ввода строки
	Клавиша **Del** используется для удаления символа, находящегося под курсором
	Клавиша **Ins** (Insert – вставка) предназначена для переключения между двумя режимами ввода символов
	**Backspace** (стрелка влево над клавишей **Enter**) удаляет символ, слева от курсора
	**Клавиши перемещения курсора** предназначены, как правило, для перемещения курсора или «перелистывания» на экране текста
	Клавиша **NumLock** (блокировка цифр) предназначена для переключения между двумя режимами работы цифровой клавиатуры
	Клавиша **Esc** (escape – убегать, спасаться), как правило, используется для отмены какого-либо действия, выхода из режима программы и т. д.
	Функциональные клавиши предназначены для различных специальных действий. Их действие определяется выполняемой программой
	Клавиша **Shift** для ввода прописных букв (режим **CapsLock** отключен)
	**CapsLock** включает и выключает режим ввода прописных букв
	**Control** (**Ctrl**) и **Alt** вводятся в комбинации с другими клавишами, и программа может особым образом реагировать на такие комбинации
	Клавиша **Print Screen** предназначена для распечатки текстового (графического) экрана
	Клавиша **Scroll Lock –** служебная клавиша для фиксации регистра

Имеются комбинации клавиш, обрабатываемые специальным образом. Некоторые из них показаны в табл. 2. Комбинация клавиш означает, что необходимо сначала нажать первую клавишу, не отпуская ее, нажать вторую и т. д.

Таблица 2

Control Break	Завершение (прерывание) работы выполняемой программы или команды
Control Alt Delete	Перезагрузка DOS
Control C	Прекращение работы любой программы или команды MS DOS (для DOS)
Control S	Приостановление выполнения программ (для DOS)
Control Num Lock	Приостановление выполнения программ
Control PrtSc	Включение и выключение режима копирования на принтер выводимой на экран информации
Control P	Включение и выключение режима копирования на принтер выводимой на экран информации (для DOS)

2.1.5. Печатающие устройства

Печатающие устройства предназначены для вывода информации на бумагу [4, 6]. Исторически первыми печатающими устройствами для ЭВМ были **литерные принтеры** с ограниченным набором печатаемых символов. Как правило, в настоящее время такие принтеры не применяются. Современные принтеры отличаются способами нанесения красителя на бумагу.

Матричные принтеры. Принцип печати этих принтеров таков: печатающая головка принтера содержит вертикальный ряд тонких металлических стержней (иголок). Головка движется вдоль печатаемой строки, а стержни в нужный момент ударяют по бумаге через красящую ленту. Это и обеспечивает формирование на бумаге символов и изображений.

Струйные принтеры. В этих принтерах изображение формируется микрокаплями специальных чернил, выдуваемых на бумагу, с помощью сопел. Этот способ печати обеспечивает более высокое качество печати по сравнению с матричными, он очень удобен для цветной печати.

Лазерные принтеры. В этих принтерах для печати используется принцип ксерографии: изображение переносится на бумагу со специального барабана, к которому электрически притягиваются частички краски, при этом печатающий барабан электризуется с помощью лазера по командам из компьютера. Лазерные принтеры обеспечивают в настоящее время наилучшее качество и скорость печати.

Для вывода чертежей на бумагу в системах конструирования систем автоматизированного проектирования (САПР) используют ***графопостроитель*** **(*плоттер*)**. Плоттеры бывают барабанного типа (работают с рулоном бумаги) и планшетного типа (в них лист бумаги лежит на плоском столе).

2.1.6. Ручные манипуляторы

Ручные манипуляторы или координаторные устройства предназначены для облегчения перемещения курсора по экрану, изменения рабочей среды, выбора действия, отметки блоков информации, рисования объектов и т. п. [5].

Наиболее популярный тип координатного устройства – манипулятор типа ***мышь*** (Mouse). Мышь представляет собой легко умещающуюся в ладони коробочку с кнопками и проводом. Перемещение мыши по столу вызывает перемещение курсора мыши по экрану, а команды передаются нажатием кнопок. В символьном режиме экрана курсор мыши по умолчанию имеет вид прямоугольника, в графическом – вид стрелки, направленной влево вверх. Наиболее распространены механические мыши с шариком в основании. Самые точные и надежные мыши – оптические, построенные на основании светового индикатора. Известны также «безхвостные» мыши с автономным питанием и радиопередачей сигналов. Помимо манипуляторов типа мышь используются ***шаровые манипуляторы*** (***trackboll***) и ***джойстики*** (***joystick***).

2.1.7. Устройства ввода изображений

Устройства ввода изображений предназначены для считывания графической информации в компьютер [5]. Для ввода изображения наиболее часто используется ***сканер*** (***scanner***).

Сканер создает в компьютере электронную копию изображения. Изображение может быть текстом, рисунком, фотографией, диаграммой, проекцией трехмерного предмета на плоскость или чем-нибудь другим. Оно считывается многоэлементными фотоприемными линейками с использованием протяженного осветителя и объектива. Существуют: ручные сканеры (оператор сам должен перемешать ручной сканер

по изображению); планшетные сканеры (для считывания изображения в него кладется лист бумаги или целая пачка при наличии автоподатчика); роликовые сканеры (сами протягивают бумагу сквозь себя); проекционные сканеры (обеспечивают ввод как документов, так и проекций трехмерных объектов). Кроме ввода графической информации, в последнее время сканеры стали использовать для ввода текстовой информации с переводом ее в формат ASCII. Для этого используют аппаратные и программные средства оптического распознавания символов. В системах автоматического конструирования для ввода чертежей в компьютер используется ***графический планшет***.

Для ручного создания рисунка можно использовать такое устройство, как ***диджитайзер*** (***digitizer – оцифровщик***). Фактически – это планшет для рисования специальным пером. В настоящее время ведутся активные разработки в области перьевого ввода текста, но о повсеместном применении pen-распознования говорить пока рано.

2.1.8. Коммуникационное оборудование

Связь между ЭВМ необходима для обмена информацией с другими компьютерами, кроме того, широко применяется управление удаленными компьютерами. В зависимости от степени удаленности машин для обеспечения компьютерной коммуникации необходимо различное оборудование [5, 6].

Связь между двумя компьютерами, находящимися в одном месте, осуществляется посредством кабеля, которым нужно соединить последовательные или параллельные порты. Для совместного функционирования многих компьютеров в системе используются ***локальные вычислительные сети*** (ЛВС). Персональный компьютер подключается к ЛВС посредством сетевого адаптера. Наиболее простые, так называемые ***одноранговые***, ЛВС предназначены для рабочих групп. Они позволяют операторам нескольких компьютеров использовать общие диски, принтеры и другие устройства, передавать друг другу сообщения и выполнять другие коллективные операции. В ***многоранговых*** ЛВС для хранения разделяемых данных и программ используются серверы – более мощные компьютеры, чем машины пользователя. В последнее время большое значение приобретает объединение ЛВС для совместного функционирования в так называемых ***сетях RAN*** и ***WAN – Regional Area Network*** и ***Wide Area Network***.

Для передачи данных на действительно большие расстояния используются линии телефонной сети и выделенные линии. Преобразование цифровой информации в сигналы речевого диапазона (модуляцию) и звуковых сигналов в цифровую информацию (демодуляцию) выпол-

няет ***модем*** – модулятор-демодулятор (рис. 9). Используются также ***факс-модемы –*** устройства, сочетающие возможности модема и средства для обмена факсимильными изображениями с другими факс-модемами и обычными телефаксными аппаратами (рис. 10).

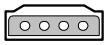


Рис. 9. Модем

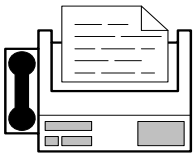


Рис. 10. Факс-модем

2.2. Операционная система

2.2.1. Начальные сведения об операционной системе

Операционная система – это программа, которая загружается при включении компьютера. Она производит диалог с пользователем, осуществляет управление компьютером, его ресурсами (оперативной памятью, местом на дисках и т. д.), запускает другие (прикладные) программы на выполнение. Операционная система обеспечивает пользователю и прикладным программам удобный способ общения (интерфейс) с устройствами компьютера. Основная причина необходимости операционной системы состоит в том, что элементарные операции для работы с устройствами компьютера и управления ресурсами компьютера – это операции очень низкого уровня, поэтому действия, которые необходимы пользователю и прикладным программам, состоят из нескольких сотен или тысяч таких элементарных операций. Операционная система скрывает от пользователя эти сложные и ненужные ему подробности и предоставляет ему удобный интерфейс для работы. Она выполняет также различные вспомогательные действия, например копирование и печать файлов. Кроме того, операционная система осуществляет загрузку в оперативную память всех программ, передает им управление в начале их работы, выполняет различные вспомогательные действия по запросу выполняемых программ и освобождает занимаемую программами оперативную память при их завершении [4].

В последнее время получили распространение следующие операционные системы: MS DOS, PC DOS, DR DOS, OS/2, Windows 3.1x, Windows 95, Windows 98, Windows 2000, Windows-NT.

2.2.2. Основные составные части операционной системы MS DOS

Операционная система состоит из следующих частей [4, 5].

***Базовая система ввода-вывода* (*BIOS*)**, находящаяся в постоянной памяти (ПП) компьютера. Эта часть операционной системы является «встроенной» в компьютер. Ее назначение состоит в выполнении наиболее простых и универсальных услуг операционной системы, связанных с осуществлением ввода-вывода. Базовая система ввода-вывода содержит также тест функционирования компьютера, проверяющий работу памяти и устройств компьютера при включении его электропитания. Кроме того, базовая система ввода-вывода содержит программу вызова загрузчика операционной системы.

***Загрузчик операционной системы* –** это программа, функция которой заключается в считывании в память модулей операционной системы, которые и завершают процесс загрузки.

***Дисковые файлы* –** два файла (*IO.SYS* и *MSDOS.SYS*), которые загружаются в память загрузчиком операционной системы и остаются в памяти компьютера постоянно. Файл *IO.SYS* – дополнение к базовой системе ввода-вывода. Файл *MSDOS.SYS* реализует основные высокоуровневые услуги DOS.

Командный процессор MS DOS обрабатывает команды, выводимые пользователем. Командный процессор находится в дисковом файле *COMMAND.COM* на диске, с которого загружается операционная система. Некоторые команды пользователя (внутренние) командный процессор выполняет сам, для выполнения остальных (внешних) команд пользователя командный процессор ищет на дисках программу с соответствующим именем и если находит ее, то загружает в память и передает ей управление. По окончании работы программы командный процессор удаляет программу из памяти и выводит сообщение о готовности к выполнению команд.

***Внешние команды MS DOS* –** это программы, поставляемые вместе с операционной системой в виде отдельных файлов. Эти программы выполняют действия обслуживающего характера.

***Драйверы устройств* –** это специальные программы, которые дополняют систему ввода-вывода MS DOS и обеспечивают обслуживание новых устройств или нестандартное использование имеющихся устройств.

2.2.3. Основные понятия операционной системы MS DOS

Как ранее было упомянуто, операционная система предназначена для осуществления управления компьютером, его ресурсами (оперативной памятью, местом на дисках и т. д.) и обеспечения пользователю и прикладным программам удобного способа общения (интерфейс) с устройствами компьютера.

Программа (Program). Последовательность операций, выполняемых вычислительной машиной для реализации какой-либо задачи, часто называют программным обеспечением, или прикладной программой. Программы пишутся на специальных машинных языках и хранятся в файлах. Выделяют резидентные программы, которые после завершения работы оставляют в памяти машины какую-либо свою часть, которая позволяет либо осуществлять постоянно заданные функции, либо вызывать в любой момент, в том числе и во время выполнения другой программы, данную программу.

Файл (File). Это поименная область на диске или другом машинном носителе, содержащая определенную совокупность данных. В файлах могут храниться тексты программ, документы, готовые к выполнению программы, и т. д. Использование понятия файла существенно облегчает работу пользователя, т. к. в этом случае он не обязан знать, в каком физическом порядке и где именно находятся его данные. Чтобы различить файлы, каждый имеет собственное имя, причем компьютеру безразлично, какое имя носит тот или иной файл, поскольку он получает от операционной системы инструкции низкого уровня.

Имя файла (Filename). Принято, что каждое название файла состоит из двух частей: собственно имени файла (filename) и расширения (extension). Имя файла имеет длину от 1 до 8 символов. В имени файла можно использовать любые символы из следующих: A...Z a...z 0...9 ’ ` ! @ # $ % ^ & ( ) _ - { }. Эти символы можно применять как в имени файла, так и в его расширении. В русифицированных версиях MS DOS можно применять русские буквы в именах файлов. Расширение имени файла (extension) состоит из точки, за которой следует от одного до трех символов. Использование расширения является необязательным. Оно, как правило, описывает содержимое файла, поэтому использование расширения весьма удобно. По расширению можно узнать, какая программа создала файл. Например:

.com, .exe	готовые к выполнению программы

.bat	командные (Batch) файлы
.pas	программы на Паскале
.arj, .rar, .zip	архивные файлы
.bak	копия файла, делаемая перед его изменением

Каталог (Directory). Каталог – это специальное место на диске, в котором хранятся имена файлов, сведения о размере файлов, времени их последнего обновления, атрибуты (свойства) файлов и т. д. Если в каталоге хранится имя файла, то говорят, что этот файл находится в данном каталоге. Все каталоги, кроме корневого, на самом деле являются файлами специального вида. Каждый каталог имеет имя, и он может быть зарегистрирован в другом каталоге. Требования к именам каталогов те же, что и к именам файлов. На каждом магнитном диске имеется один главный, или корневой, каталог. В нем регистрируются файлы и подкаталоги. В каталогах 1-го уровня регистрируются файлы и каталоги 2-го уровня и т. д. Получается иерархическая древообразная структура каталогов на магнитном диске (см. рис. 11).

Дисковод (Disk Drive). Под дисководом понимают устройство, которое вращает диск и обеспечивает перемещение головок при записи/чтении. Иногда этим именем называют и накопитель на магнитных дисках. Приводы гибких дисков обозначаются как A и B, жесткого – как C.

В операционной системе MS DOS можно разделить жесткий диск на несколько частей и работать с ними как с отдельными дисками. Эти диски называются логическими дисками, или разделами жесткого диска. Каждый логический диск имеет имя (букву), по которому к нему можно обращаться (C: , D: , E: , F: ...).

Команда (Command). Под командой понимают как команду, которую пользователь вводит с клавиатуры, так и программу, которая эту программу исполняет. Работа с MS DOS сводится к выполнению программ – прикладных программ пользователя и сервисных программ MS DOS, в том числе встроенных в командный процессор. Программы обоих типов вызываются командами, которые дает пользователь. Чтобы дать команду, необходимо набрать ее текст после приглашения MS DOS и нажать кнопку ввода – **Enter**. Все команды MS DOS можно разделить на внутренние и внешние команды. Внутренние команды MS DOS встроены непосредственно в командный процессор (***COMMAND.COM***). Программы, находящиеся на магнитных дисках в виде COM- и EXE-

файлов, вызываются на выполнение посредством внешних команд MS DOS.

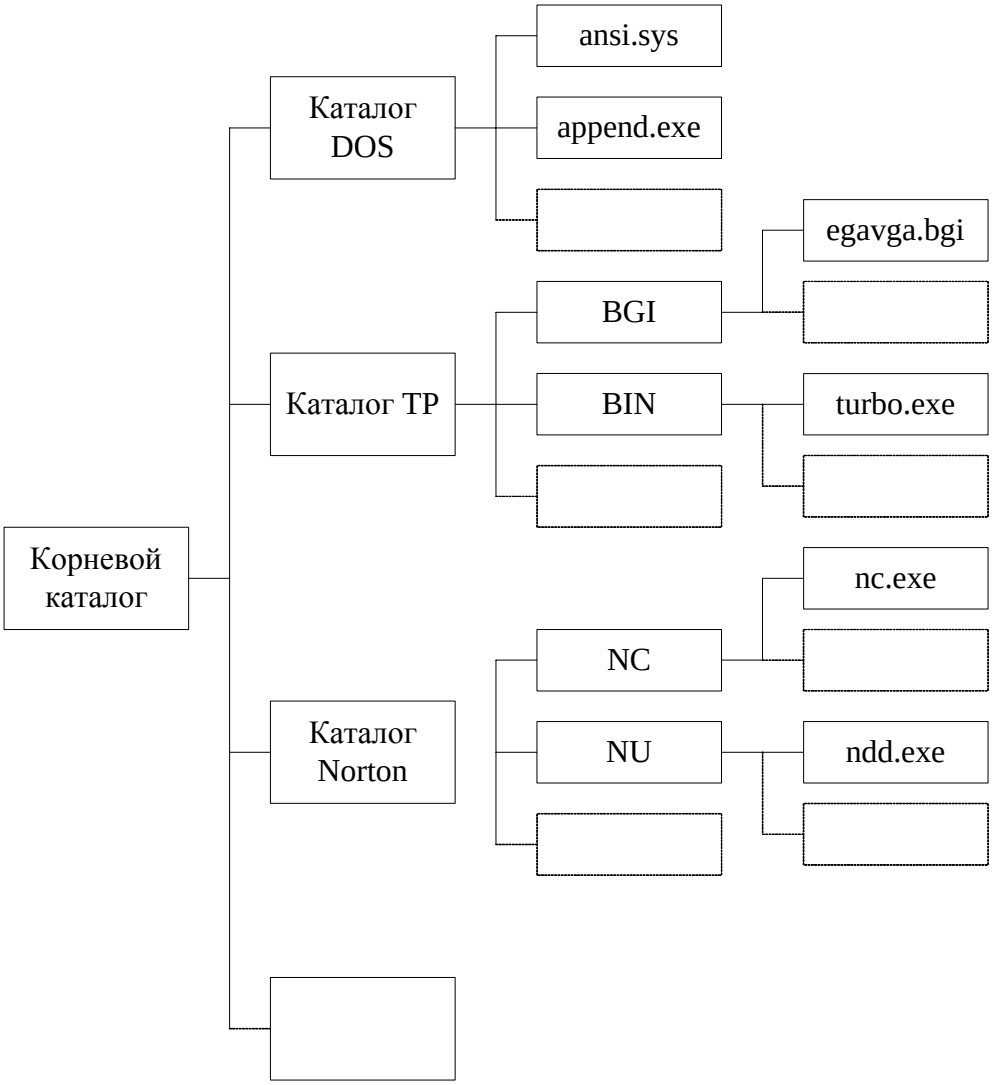


Рис. 11. Пример файловой системы на магнитном диске

Кроме того, внешней командой можно вызывать файлы с расширением BAT, которые являются не программами, а списками команд MS DOS.

В табл. 3 приведены некоторые внутренние команды MS DOS.

Таблица 3

Команда	Назначение
DATE	Вывод на экран или установка даты
TIME	Вывод или установка системного времени
CLS	Очистка экрана
DIR	Вывод содержимого каталога
COPY	Копирование файла в другое место
MD	Создание каталога
DEL	Удаление одного или нескольких файлов

2.3. Программы-оболочки

Основное назначение программ-оболочек – облегчить пользователю работу с файлами на диске, запуск программ, доступ к функциям управления файлами, каталогами и дисками, и, кроме того, они предоставляют пользователю ряд дополнительных возможностей. Рассмотрим особенности работы в некоторых из них [4–6].

2.3.1. Norton Commander

Оболочка Norton Commander (NC) была выпущена в 1986 г. фирмой Peter Norton Computing [4, 6]. Программа Norton Commander позволяет выполнять большое количество различных функций, в частности:

– наглядно изображать содержание каталогов на дисках;

– изображать дерево каталогов с возможностью перехода в нужный каталог с помощью указания его на этом дереве, а также создания, переименования и удаления каталогов;

– удобно копировать, переименовывать, пересылать и удалять файлы;

– просматривать текстовые файлы, документы, сделанные с помощью различных редакторов текстов, базы данных и таблицы табличных процессоров;

– редактировать текстовые файлы;

– выполнять любые команды DOS;

– изменять атрибуты файлов.

Запуск Norton Commander осуществляется вводом команды ***NC.EXE***. Вид экрана приведен на рис. 12.

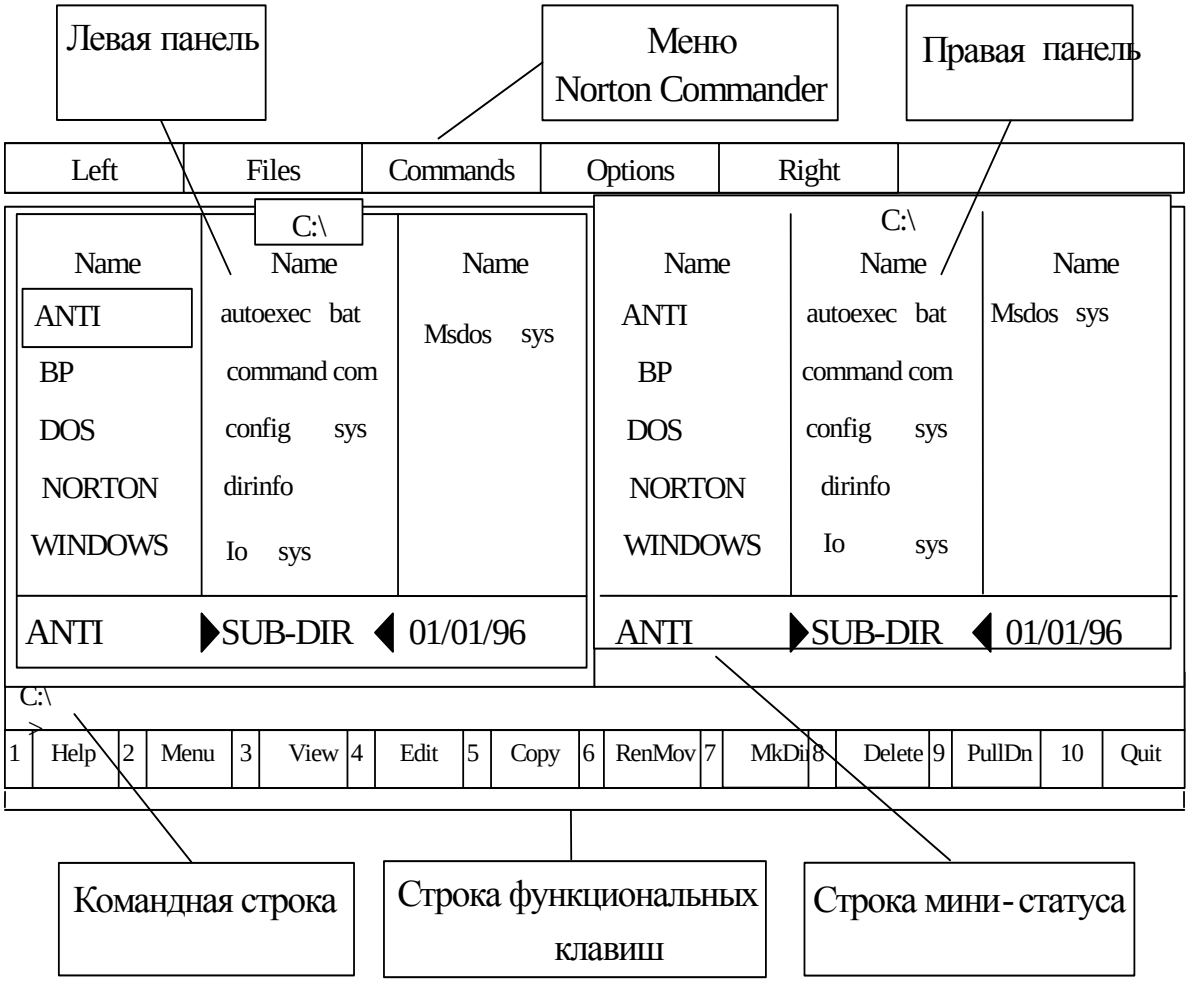


Рис. 12. Вид экрана при работе с Norton Commander

На экране можно использовать одновременно до двух панелей любого типа. Для управления панелями Norton Commander можно использовать следующие комбинации клавиш [4]:

Ctrl – O	убрать панели с экрана или вывести панели на экран
Ctrl – P	убрать одну из панелей (не текущую) с экрана/вывести панель на экран
Ctrl – U	поменять панели местами
Ctrl – F1	убрать левую панель с экрана или вывести левую панель на экран
Ctrl – F2	убрать правую панель с экрана или вывести правую панель на экран
Alt – F1	вывести в левой панели оглавление другого диска
Alt – F2	вывести в правой панели оглавление другого диска

Оглавление каталога в панели. Если в панели Norton Commander выводится оглавление каталога, то сверху панели отображается имя этого каталога. При выводе имен файлов и подкаталогов Norton Commander использует такое правило: имена файлов выводятся строчными буквами, а имена подкаталогов – прописными.

Клавишей **Tab** можно перевести выделенный участок на другую панель Norton Commander.

Информационная панель. В панели Norton Commander можно вывести сводную информацию о диске и каталоге на другой панели. Наверху информационной панели выводится строка ***Info***. В панели изображаются следующие сведения:

– емкость оперативной памяти компьютера в байтах (...*Bytes Memory*);

– количество свободной оперативной памяти в байтах (...*Bytes Free*);

– емкость текущего дисковода в байтах (... *bytes on drive* ...);

– количество свободного места на текущем диске (... *bytes free on drive*...);

– количество файлов в каталоге, выведенном на другой панели Norton Commander, и их общий размер в байтах (... *files use* ... *bytes in* ..).

Использование функциональных клавиш. В нижней строке экрана Norton Commander выводит напоминание о значении функциональных клавиш [5]. Ниже кратко описывается их назначение:

F1 – *Help* – краткая информация о назначении клавиш при работе с Norton Commander;

F2 – *Menu* – запуск команд, указанных в списке, заданном пользователем (пользовательское меню);

F3 – *View* – просмотр файла. Можно просматривать текстовые файлы, базы данных, таблицы табличных процессоров, рисунки графических редакторов, просматривать содержимое архивных файлов, проигрывать звуковые WAV-файлы;

F4 – *Edit* – редактирование файла. Для редактирования может быть использован встроенный редактор Norton Commander или любой другой редактор, указанный пользователем;

F5 – *Copy* – копирование файла. В середине экрана появляется запрос о том, куда копировать файл. По умолчанию файл копируется в каталог, изображенный на другой панели. Можно набрать и другое имя файла (каталога). Затем для копирования надо нажать **Enter**, для отмены команды – **Esc**;

F6 – *Renmov* – переименование файла (каталога) или пересылка файла в другой каталог. Можно задать новое имя файла (каталога) или

имя каталога, в который надо пересылать данный файл. Для начала переименования или пересылки надо нажать **Enter**, для отмены команды – **Esc**;

F7 – *MkDir* – создание подкаталога;

F8 – *Delete* – уничтожение файла или подкаталога;

F9 – *PullDn* – вывод меню, содержащего режимы работы Norton Commander;

F10 – *Quit* – выход из Norton Commander.

Если нажать клавишу **Alt**, то последняя строка экрана изменится. В ней будет выводиться подсказка о значении комбинации клавиш (**Alt – F1**) – (**Alt – F10**). Ниже кратко описано назначение комбинаций этих клавиш:

Alt – F1 – ***Left* –** выбор диска, изображаемого на левой панели;

Alt – F2 – ***Right* –** выбор диска, изображаемого на правой панели;

Alt – F3 – ***View* –** просмотр текстового файла. Этот режим вызывается быстрее, но позволяет просматривать только текстовые файлы и не имеет некоторых возможностей, доступных при просмотре с помощью клавиши **F3**;

Alt – F4 – ***Edit* –** редактирование файла с помощью альтернативного редактора;

Alt – F7 – ***Find* –** поиск файла на диске;

Alt – F8 – ***History* –** просмотр и повторное выполнение ранее введенных команд;

Alt – F9 – ***Ega Ln* –** переключение с 25 на 43 (для монитора EGA) или 50 (для монитора VGA) строк на экране, и наоборот;

Alt – F10 – быстрый переход в другой каталог.

Работа с файлами. Norton Commander позволяет выполнять различные действия над файлами (просмотр, редактирование, копирование, переименование, пересылка, удаление, поиск файла на диске, изменение атрибутов) и группами выделенных файлов (копирование, перемещение, переименование, уничтожение, изменение атрибутов) [4]. Выбор отдельного файла осуществляется нажатием клавиши **Insert** (**Ins**). Отмена выбора – повторным нажатием клавиши. Чтобы выбрать группу файлов, необходимо нажать клавишу «+» (на цифровой клавиатуре) и задать маску для выбора. Для отмены выбора необходимо нажать клавишу «–» (на цифровой клавиатуре) и задать маску для отмены выбора.

Просмотр файлов. При нажатии клавиши **F3** Norton Commander позволяет просматривать выделенный курсором файл. При просмотре текстовых файлов и документов имеются следующие дополнительные возможности:

– **F2** – переносить или нет на другую строку длинные строки документов;

– **F4** – вывод файла в шестнадцатеричном виде и выход из этого режима;

– **F7** – поиск подстроки в документе. Нужную подстроку нужно ввести в ответ на запрос;

– **Shift – F7** – повторение поиска той же подстроки в документе;

– **F8** – выбор режима просмотра документа;

– **F10** – выход из режима просмотра.

Редактирование файла. Для редактирования выделенного курсором файла следует нажать **F4**, если используется встроенный редактор, или **Alt – F4**, если используется альтернативный редактор. Опишем возможности встроенного редактора Norton Commander.

Курсор. Курсор (мигающий символ на экране) указывает на текущую позицию в тексте. Все изменения в тексте и вставки нового текста происходят в той позиции, на которую показывает курсор.

Перемещение курсора по тексту. Курсор можно перемещать с помощью клавиш управления курсором **(← → ↑ ↓)** на одну позицию влево, вправо, вверх и вниз. Кроме того, курсор можно перемещать по тексту с помощью следующих клавиш:

– **PgUp** и **PgDn** на страницу (размер экрана) вверх и вниз;

– **Home** и **End** на начало и конец редактируемого файла;

– **Ctrl – ←** и **Ctrl – →** на слово влево и вправо.

Ввод текста. Для ввода текста нужно переместить курсор в то место, в которое надо вводить новый текст, и начать набор текста, нажимая соответствующие буквенно-цифровые клавиши. Для окончания строки надо нажать клавишу **Enter**.

Ввод символов из верхнего регистра клавиатуры. Если необходимо ввести символ из верхнего регистра клавиатуры, нужно нажать клавишу **Shift** и, не отпуская ее, нажать клавишу с нужным символов.

Удаление символов и строк. Для удаления символов и строк можно использовать следующие клавиши:

– **Delete** или **Del** – удаление символа под курсором;

– **BackSpace** – удаление символа слева от курсора;

– **Ctrl – Y** – удаление строки;

– **Ctrl – K** – удаление текста от текущей позиции курсора до конца строки.

Назначение функциональных клавиш. При редактировании файла с помощью встроенного в Norton Commander редактора можно использовать следующие функциональные клавиши:

– **F2** – сохранить отредактированный файл;

– **Shift – F2** – сохранить отредактированный файл под другим именем (новое имя запрашивается);

– **F10** – выйти из режима редактирования (иной способ – нажатие клавиши **Esc**);

– **Shift – F10** – сохранить отредактированный файл и выйти из режима редактирования;

– **F7** – поиск подстроки в документе. Нужная подстрока вводится в ответ на запрос;

– **Shift – F7** – повторный поиск той же подстроки в документе;

– **F1** – вывод на экран справки о назначениях клавиш при редактировании.

Копирование файла (файлов). Для копирования файлов надо выделить нужный файл или выбрать группу файлов и нажать клавишу **F5**. В центре экрана появится запрос о том, куда надо копировать файл или файлы. В ответ на запрос можно:

– ввести имя каталога, в который надо копировать файл (файлы);

– ввести новое имя файла (если копируется более одного файла, то в этом имени должны быть символы * или ?);

– нажав клавишу **F10**, вывести на экран дерево каталогов текущего диска и выбрать в нем каталог, в который надо скопировать файл (файлы). Для выбора каталога надо выделить его с помощью клавиш перемещения курсора и затем нажать **Enter**.

После этого надо нажать клавишу **Enter** для начала копирования файлов или **Esc** для отмены копирования. Если на появившийся запрос просто нажать клавишу **Enter**, то файлы будут скопированы в каталог на другой панели с теми же именами.

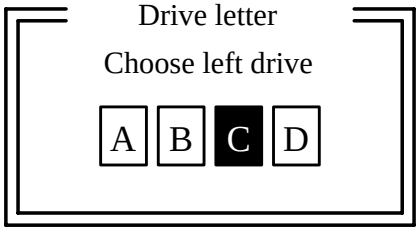
Переименование и пересылка файлов. Для переименования или пересылки в другой каталог файлов с помощью NC надо выделить переименовываемый или пересылаемый файл или выбрать группу файлов и нажать клавишу **F6**. Если файл пересылается в другой каталог того же диска, то содержимое файла остается на том же месте диска, а в другой каталог включается только ссылка на этот файл (т. е. элемент каталога), а из исходного каталога эта ссылка исчезает. Если же файл пересылается на другой диск, то он просто копируется на него и после успешного окончания копирования исходный файл уничтожается.

Удаление файлов. Для удаления файлов с помощью NC надо выделить нужный файл или выбрать группу файлов и нажать клавишу **F8**. При удалении файлов на экран выводится запрос на подтверждение удаления. Для подтверждения удаления нужно нажать клавишу **Enter**, для отмены – **Esc**.

Работа с каталогами. Создание каталога. Для создания каталога следует нажать клавишу **F7**. NC выведет на экран запрос об имени каталога (*Create the directory*). Необходимо ввести это имя и нажать **Enter**.

Удаление каталога. Для удаления каталога следует указать его курсором и нажать клавишу **F8**.

Переход на другой диск. Для того чтобы в панели NC вывести оглавление другого диска, следует нажать **Alt – F1** – для левой панели, **Alt – F2** – для правой панели. На экран будет выведен список доступных дисков.



Затем надо выбрать имя нужного диска и нажать **Enter**. NC прочтет оглавление текущего каталога на указанном диске и выведет его на экран.

Меню Norton Commander. С помощью меню Norton Commander можно установить наиболее удобный вид представления информации на экране, изменить режимы работы NC, а также выполнять некоторые другие действия [4, 5].

Работа с меню. Для входа в меню следует нажать клавишу **F9**. В верхней строке экрана появится строка, содержащая пункты меню. Один из пунктов меню является выделенным. Выбрав нужный пункт, нажимаем клавишу **Enter** и под ним откроется соответствующее ему подменю. Выбрав нужный пункт подменю, нажимаем клавишу **Enter**.

Выход из меню. Для выхода из меню или подменю Norton Commander следует использовать клавишу **Esc**.

Пункты меню ***Left*** и ***Right*** задают режимы вывода информации соответственно в левой и правой панелях Norton Commander.

Пункт меню ***Files*** дает возможность производить различные операции с файлами. Многие из этих операций закреплены за функциональными клавишами.

Пункт меню ***Commands*** позволяет выполнять различные команды Norton Commander.

Пункт меню ***Options*** позволяет задавать конфигурацию и режимы работы Norton Commander и указывать, какой редактор будет использоваться при редактировании файлов.

Меню команд пользователя. При нажатии клавиши **F2** Norton Commander выводит на экран список команд, указанный пользователем

в файле ***NC.MNU***., после чего можно выполнить соответствующие команды из списка.

2.3.2. Microsoft Windows

Операционная система Windows 98 является, по сути, эволюционным продолжением Widows 95, которая полностью изменила принцип управления файлами и интерфейс, что сделало операционную систему доступной даже для новичков.

Windows 98 представляет собой высокопроизводительную, многозадачную и многопотоковую 32-разрядную операционную систему с графическим интерфейсом и расширенными сетевыми возможностями, работающую в защищенном режиме [7 – 9].

Запуск Windows 98 осуществляется автоматически при включении компьютера. После загрузки на экране появится рабочий стол Windows (см. рис.13).

Windows 2000 – новая операционная система Microsoft, основанная на технологии Windows NT, создана группой разработчиков под руководством Дэйва Катлера. Windows 2000 – полностью 32-разрядная ОС с приоритетной многозадачностью и улучшенной реализацией работы с памятью. В основе проекта лежат те же принципы, которые когда-то обеспечили успех NT. Имеют место следующие виды поставок: Windows 2000 Professional, Windows 2000 Server, Windows 2000 Advanced Server и Windows 2000 DataCenter Server. Отличаются они друг от друга, во-первых, количеством служб и программ, входящих в поставку, во-вторых, степенью поддержки аппаратного обеспечения.

Windows 2000 представляет собой операционную систему нового поколения для делового использования на самых разнообразных компьютерах — от переносных компьютеров до высококлассных серверов, является наилучшей операционной системой для ведения коммерческой деятельности в интернете. Это наилучшая операционная система, которая позволяет применять любое новейшее оборудование – от самых маленьких мобильных устройств и до самых больших серверов.

Работа с манипулятором мышь. При работе с операционной системой Windows большую часть операций можно осуществлять с помощью компьютерной мыши. Компьютерная мышь является манипуляторным устройством, позволяющим общаться с компьютером через операционную систему. Мышь на экране управляет указателем (курсором), вид которого зависит от выполняемой операции. Движение мыши по ровной поверхности автоматически повторяется соответствующим перемещением указателя на экране. В Windows 98 используются две клавиши мыши:

– ***левая клавиша*** – для выполнения действий (основная);

– ***правая клавиша*** – для получения информации по свойствам любого объекта.

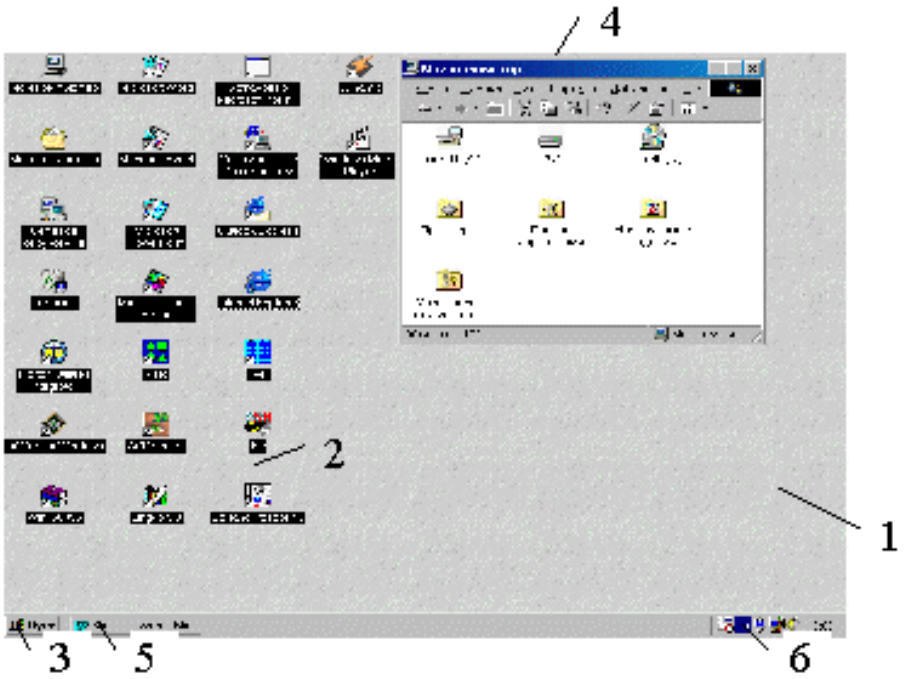


Рис. 13. Рабочий стол Windows:
1 – рабочий стол; 2 – папки и значки; 3 – кнопка **Пуск**;
4 – открытое окно; 5 – кнопка свернутого окна приложения;
6 – значки утилит, запускаемых в фоновом режиме

С помощью мыши пользователь может выполнить следующие операции [7, 8]:

– ***Щелчок*** (***фиксация***). Для выполнения необходимо быстро нажать и отпустить кнопку мыши в тот момент, когда вершина стрелки указателя мыши неподвижна и находится на нужном объекте. Одна из самых распространенных операций, выполняемая щелчком мыши – выбор объекта.

– ***Двойной щелчок*** производится быстрым двойным нажатием на кнопку мыши без ее перемещения. Двойной щелчок выполняют после того, как указатель мыши помещен на объект (элемент), при этом щелкнуть надо быстро, иначе система воспримет их как два одиночных щелчка и выполнит другие команды. Двойной щелчок используется при открытии документов, запуске программ.

– ***Перетаскивание*** (***перемещение***) объекта производится после установки указателя мыши на нужном объекте (элементе). Затем, нажав левую кнопку мыши и не отпуская ее, перемещают мышь, двигая объект или элемент к конечному положению. Кнопку мыши отпускают в тот момент, когда хотят закончить операцию. Перетаскивание широко при-

меняется для перемещения значка папки или файла, окна или его границы.

Использование меню «Пуск». Меню «**Пуск**» является основным рабочим инструментом для запуска программ. Чтобы активизировать меню «**Пуск**», необходимо курсор мыши подвести к кнопке «**Пуск**» и нажать левую клавишу мыши. На экране появится меню «**Пуск**». После этого, перемещая курсор мыши вдоль пунктов меню, можно раскрывать соответствующие подменю (рис. 14).

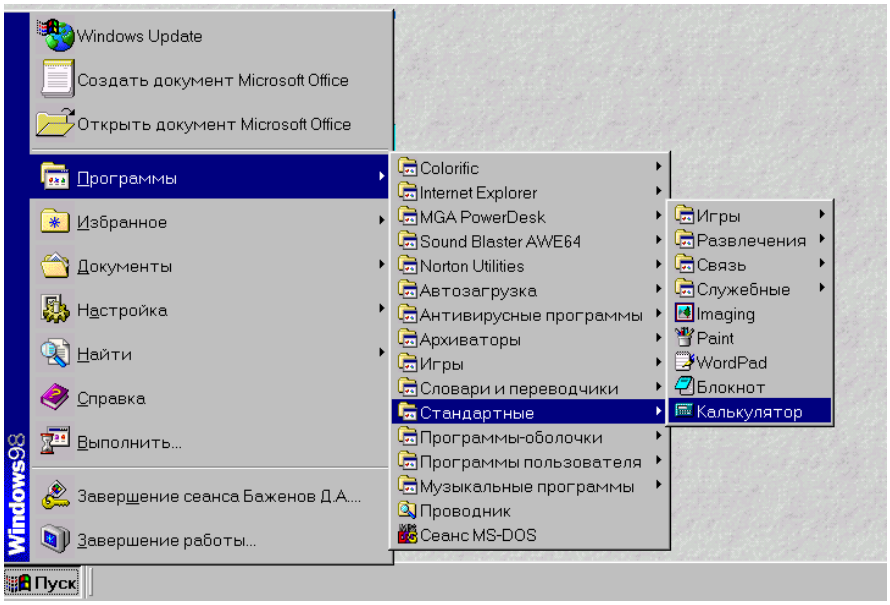


Рис. 14. Меню «Пуск»

Когда нужное имя программы, документа или команды будет подсвечено, для запуска или открытия необходимо еще раз нажать левую кнопку мыши. Можно запустить программу, документ или команду из меню и без использования мыши. Для этого необходимо нажать комбинацию клавиш **Ctrl – Esc**, а далее перемещение по пунктам меню осуществляется с помощью клавиш перемещения курсора (←, ↑, →, ↓). Для запуска или открытия необходимо нажать клавишу **Enter**. Например, запустите программу «Калькулятор» (***Пуск–Программы–Стандартные–Калькулятор***). На экране появится программа «Калькулятор» (рис. 15).

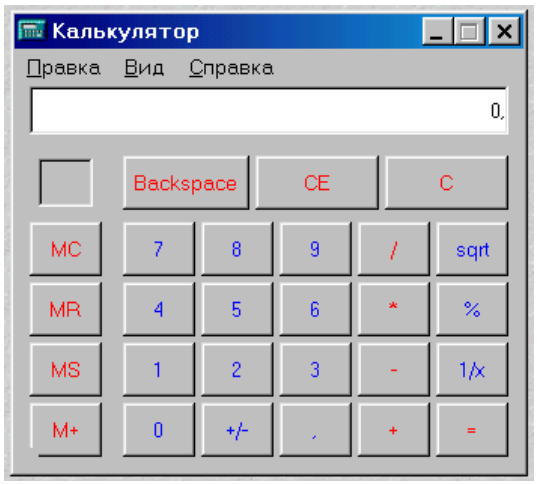


Рис. 15. Программа «Калькулятор»

Для закрытия программы подведите указатель мыши к кнопке закрытия окна (см. рис. 16) и нажмите левую кнопку мыши или закройте программу с помощью комбинаций клавиш **Alt –F4**.

Использование ярлыков на рабочем столе. Ярлыки – это значки быстрого доступа к программам и документам. Двойным щелчком мыши на ярлыке рабочего стола вы запускаете соответствующую программу и затем загружаете нужный документ (в Windows 98 существует возможность запуска программ одним щелчком мыши при использовании активного рабочего стола). Чтобы запустить программу или открыть документ с помощью ярлыка, необходимо подвести указатель мыши к необходимому значку на рабочем столе и два раза быстро нажать левую кнопку мыши.

Соответствующая программа или документ откроются автоматически. Аналогично открываются и окна. Например, найдите на рабочем столе ярлык «***Мой компьютер***» и откройте его (см. рис. 16).

Существует несколько способов открытия окна:

– щелкнуть на кнопке «**Пуск**», а затем на имени команды, меню, папки или документа;

– щелкнуть на нужном вам ярлыке на рабочем столе;

– в окне «***Мой компьютер***» щелкнуть на значке программы или документа;

– щелкнуть на кнопке панели, например панели Microsoft Office.

Переключение между окнами [7]. Одновременно вы можете запускать несколько программ, и окна каждой из них будут располагаться

на рабочем столе, но только одно окно из всех будет активным. Если вы хотите работать в окне, его строка заголовка должна быть выделена.

Активное окно отличается от других цветом строки заголовка, – как правило, более темным, а в других окнах строки заголовков светлее. Если окна перекрываются, активное окно будет располагаться поверх других, а соответствующая ему кнопка на панели задач будет выглядеть нажатой.

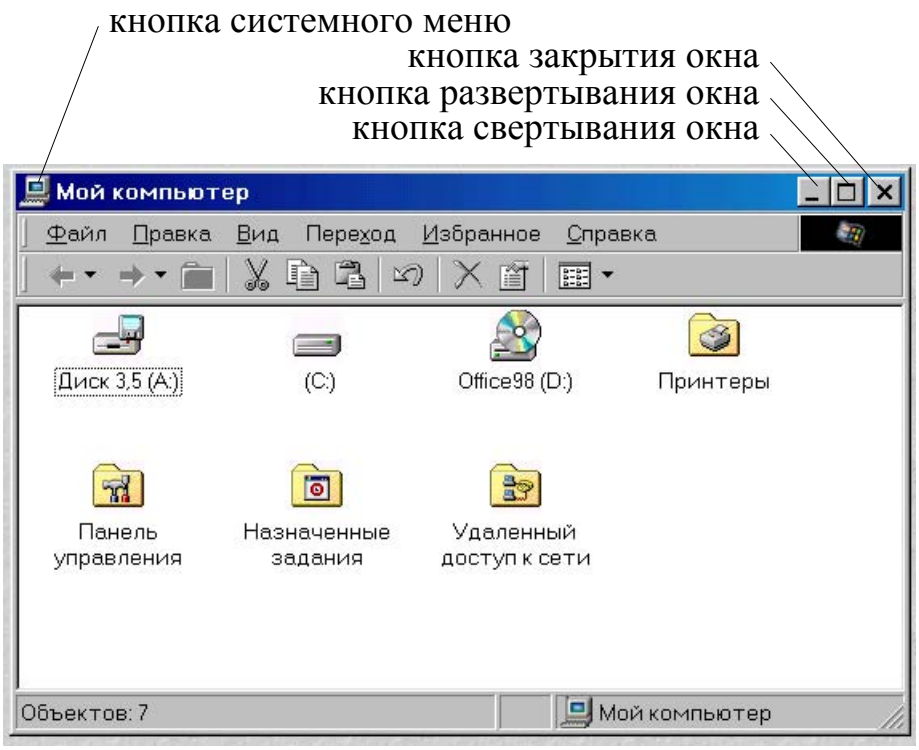


Рис. 16. Окно «*Мой компьютер*»

Вы можете выбрать окно и таким образом переключиться с одной программы на другую. Это можно сделать с помощью следующих действий:

– щелкнуть по кнопке окна на панели задач;
– щелкнуть по строке заголовка окна;
– щелкнуть по любой видимой части окна;
– удерживая клавишу **Alt**, нажимать клавишу **Tab** до тех пор, пока не будет выделено окно нужной вам программы. Когда вы окажетесь в нужном вам окне, отпустить клавишу **Alt**.

Закрытие окна. Существует несколько способов закрытия окна:

– щелкнуть по кнопке закрытия окна в верхнем правом углу окна (рис. 16);

– щелкнуть на значке в верхнем левом углу окна и выбрать опцию «***Закрыть***»;

– раскрыть меню «***Файл***» и выбрать команду ***Exit*** (Выход, Закрыть и т. п.);

– щелкнуть правой кнопкой мыши по кнопке окна в панели задач и выбрать команду «***Закрыть***»;

– нажать комбинацию клавиш **Alt – F4**.

Свертывание и развертывание окна. В Windows существует возможность свертывания и развертывания окна. Существует несколько способов свернуть и развернуть окно:

– щелкнуть по кнопке свертывания или развертывания в верхнем правом углу окна;

– щелкнуть по значку в верхнем левом углу окна и из появившегося системного меню выбрать команду «***Свернуть***» или «***Развернуть***» (рис. 17);

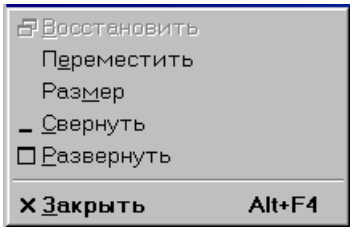


Рис. 17. Системное меню

– щелкнуть правой кнопкой мыши по кнопке окна в панели задач и выбрать из появившегося системного меню команду «***Свернуть***» или «***Развернуть***» (см. рис. 18).

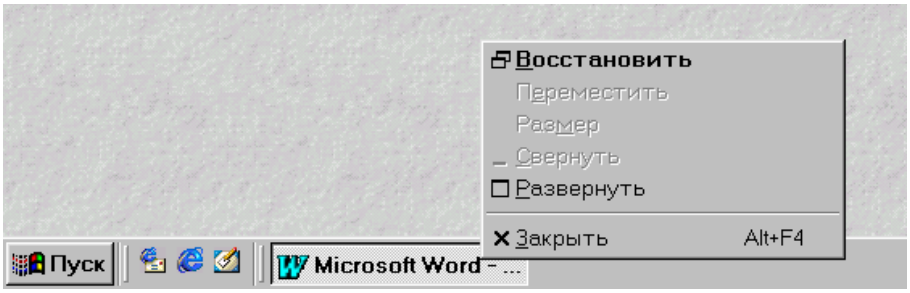


Рис. 18. Системное меню в панели задач

Перемещение окна. Самый простой способ переместить окно на рабочем столе – перетащить его с помощью мыши. Для этого необходимо поместить указатель мыши на строку заголовка окна, щелкнуть левой кнопкой мыши и, удерживая ее нажатой, перетащить окно на

нужное место и затем отпустить кнопку мыши. Можно также переместить окно с помощью клавиатуры. Для этого необходимо нажать комбинацию клавиш **Alt – пробел.** Из системного меню в левом углу окна выбрать команду «***Переместить***» и с помощью клавиш управления курсором поместить окно в нужное место. Завершив перемещение, нажать клавишу **Enter**.

Изменение размеров окна. Для изменения размеров окна необходимо поместить указатель мыши в какой-либо угол или на одну из границ окна (указатель мыши при этом приобретает вид двунаправленной стрелки), щелкнуть левой кнопкой мыши и, удерживая ее нажатой, перемещать указатель мыши до придания окну желаемых размеров, затем отпустить кнопку мыши. Можно изменять размеры окна с помощью клавиатуры. Для этого необходимо нажать клавиши **Alt – пробел**, чтобы открыть системное меню, выбрать команду «***Размер***» и затем с помощью клавиш управления курсором изменить размеры окна, нажать клавишу **Enter**.

Контекстное меню. Одной из самых полезных в Windows 98 является концепция контекстно-зависимых меню. Контекстное меню появляется на экране при щелчке правой кнопкой мыши. Контекстное меню предоставляет пользователю доступ к набору команд, соответствующих выполняемой в данный момент задаче. Чтобы отобразить контекстное меню на экране, необходимо поместить указатель мыши на любой пустой области рабочего стола и щелкнуть правой кнопкой мыши. В зависимости от местоположения указателя мыши в момент вызова контекстного меню его вид различается.

Окно «*Мой компьютер*». Это окно (см. рис. 16) является средством управления файлами, с помощью которого пользователь упорядочивает содержимое своего жесткого диска, подключает сетевые диски, периферийные устройства, управляет папками и файлами. Используя окно «***Мой компьютер***», пользователь может открыть окно диска или папки, файл, запустить программу, перемещать, копировать, переименовывать, удалять любые файлы или папки, получать сведения об объектах, просматривать другие составляющие своей системы, просматривать Web-страницы.

Для навигации можно использовать кнопки панели инструментов, меню или клавиатуру (см. рис. 19).

Чтобы просмотреть содержимое диска или папки, необходимо два раза щелкнуть на соответствующих значках мышью. Чтобы перейти к диску или папке более высокого уровня иерархии, необходимо щелкнуть на кнопке «***Вверх***» в панели инструментов. Чтобы вернуться к диску или папке, выбранной раньше, необходимо щелкнуть в панели

инструментов на кнопке «***Назад***». Чтобы выбрать диск или папку, открытую раньше, необходимо щелкнуть на кнопке со стрелкой вниз, расположенной около кнопки «***Назад***», и выбрать нужный элемент из списка. Если к каким-то папкам и дискам осуществлялся переход с помощью кнопки «***Назад***», то с помощью кнопки «***Вперед***» можно проделать обратный путь. Перейти к любому диску или папке можно, набрав путь в поле адреса и нажав клавишу **Enter**.

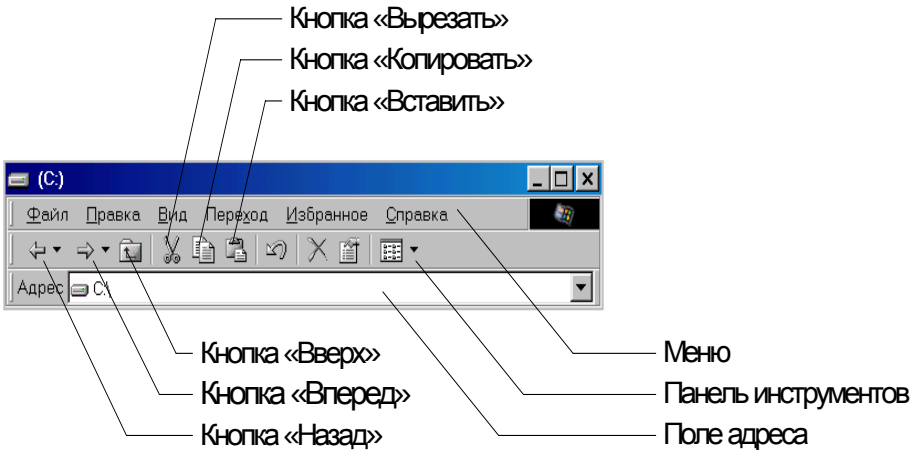


Рис. 19. Панель инструментов и меню окна «*Мой компьютер*»

Файлы и папки [7, 8]. Каждый файл на жестком диске имеет свое имя и занимает определенное место. В Windows 98 имена файлов могут содержать до 256 символов, включая пробелы и другие специальные символы. Файлы на диске хранятся в папках. Папка является аналогом каталога в операционной системе MS DOS. В каждой папке могут содержаться как файлы, так и другие папки. Точное местоположение файла называется путем к файлу, и в путь включаются все названия папок, которые нужно открыть, чтобы получить доступ к файлу. Управление файлами и папками является основой работы с компьютером. Рассмотрим некоторые средства управления файлами.

Создание папок. Создать новую папку можно с помощью окна «***Мой компьютер***». Для этого необходимо:

1. Открыть окно «***Мой компьютер***».
2. Два раза щелкнуть левой кнопкой мыши на значке диска или папки, в которой вы хотите создать новую папку.
3. Подвести курсор мыши к опции меню «***Файл***» и щелкнуть левой кнопкой.
4. В раскрывшемся меню «***Файл***» выбрать опцию «***Создать***» → «***Папку***» и щелкнуть левой кнопкой мыши.

На экране появится значок пустой новой папки. Имя этой папки – «***Новая папка***» – будет подсвечено. Необходимо назвать эту папку любым именем и нажать клавишу **Enter**.

Можно создать новую папку с помощью контекстного меню. Для этого необходимо, после того как открыта папка или диск, в которой необходимо создать новую папку, щелкнуть правой кнопкой мыши в пустой области экрана, чтобы раскрылось контекстное меню. В нем необходимо выбрать опцию «***Создать***», а затем – «***Папку***».

Переименование файлов и папок. Для переименования файлов и папок необходимо щелкнуть правой кнопкой мыши на значке папки или файла и в контекстном меню выбрать опцию «***Переименовать***». Имя файла подсветится, после чего необходимо набрать новое имя и нажать клавишу **Enter**. Существует другой способ переименования файлов и папок. Для этого необходимо выделить папку или файл с помощью щелчка левой кнопки мыши, затем подвести указатель мыши к имени и еще раз нажать левую кнопку. Имя файла подсветится, после чего необходимо набрать новое имя и нажать клавишу **Enter**.

Удаление файлов и папок. Для удаления необходимо выделить файл или папку, которую вы хотите удалить, и нажать клавишу **Delete** или же щелкнуть правой кнопкой мыши для открытия контекстного меню и выбрать в нем опцию «***Удалить***».

Восстановление удаленных файлов. Windows 98 позволяет некоторое время сохранять удаленные файлы в «***Корзине***», что позволяет их восстанавливать в случае необходимости. Для восстановления необходимо открыть корзину, щелкнув на рабочем столе на значке «***Корзина***». Когда откроется окно «***Корзина***» (рис. 20), необходимо найти имя файла или папки, которую надо восстановить. После этого необходимо щелкнуть на нужном значке правой кнопкой мыши и выбрать в контекстном меню опцию «***Восстановить***». То же можно проделать и с помощью опций меню «***Файл***» → «***Восстановить***».

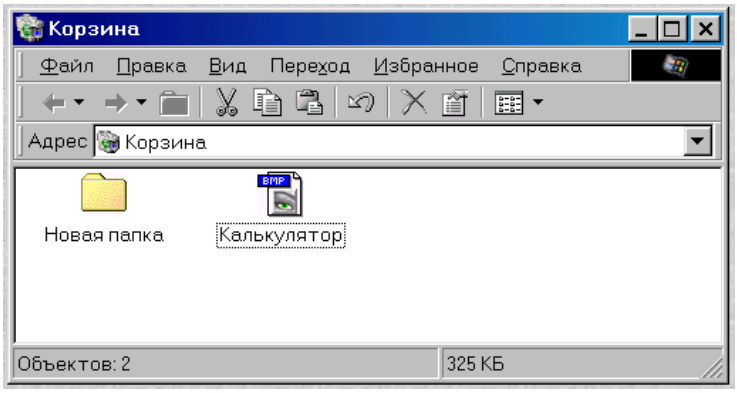


Рис. 20. Корзина

Копирование (перемещение) файлов и папок. Для копирования (перемещения) файлов и папок необходимо выполнить следующую последовательность действий.

1. В окне «***Мой компьютер***» выбрать файл или папку, которую необходимо скопировать (переместить).

Щелкнуть на нужном значке правой кнопкой мыши и выбрать команду «***Копировать***» («***Вырезать***») из появившегося контекстного меню (можно также щелкнуть левой кнопкой мыши по соответствующей кнопке в панели инструментов (см. рис. 19)).

2. Выбрать папку для копируемого (перемещаемого) объекта.
3. Щелкнуть правой кнопкой мыши на пустом месте окна выбранной папки и выполнить команду «***Вставить***» из контекстного меню (можно также щелкнуть левой кнопкой мыши по соответствующей кнопке в панели инструментов (см. рис. 19)).

Выход из Windows и выключение компьютера. Завершив работу с программами, необходимо правильно выйти из Windows, используя команду «***Завершение работы***».

1. Закрыть все прикладные программы.
2. Щелкнуть по кнопке «**Пуск**» и выбрать команду «***Завершение работы***».
3. В появившемся диалоговом окне завершения работы с Windows выбрать опцию «***Выключить компьютер***» и щелкнуть по кнопке **ОК** (рис. 21).
4. После того, как на экране появится сообщение, что теперь питание компьютера можно отключать, выключить компьютер.

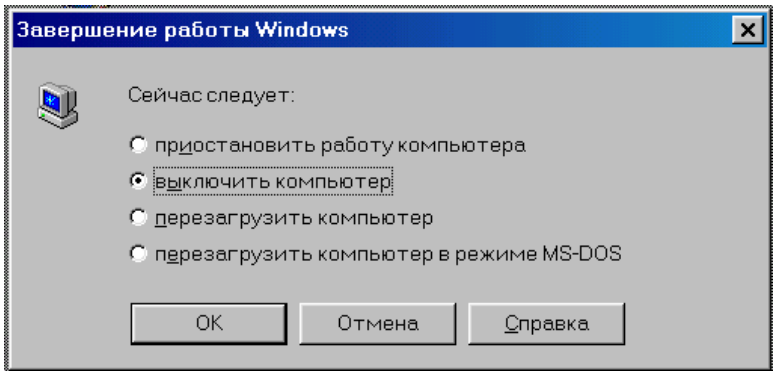


Рис. 21. Выход из Windows

3. ИНТЕГРИРОВАННАЯ СРЕДА ЯЗЫКА ТУРБО ПАСКАЛЬ

Интегрированная среда – это средство организации диалога программиста с компьютером, разработанное для облегчения написания, запуска и отладки программ на языке Паскаль [3, 10, 14].

Интегрированная среда Турбо Паскаль позволяет выполнить все этапы работы с программой:

– создать исходный текст программы на языке Паскаль;

– сохранить исходный текст программы на жёстком или гибком диске;

– отредактировать текст программы;

– выполнить программу.

Таким образом, при работе с Турбо Паскалем уже нет необходимости использовать отдельный редактор (программу для набора и исправления текстов), компилятор (программу для создания пускового файла), отладчик (программу для поиска ошибок в процессе счета) и справочник, поскольку все эти средства, а также средства работы с файлами, запуска сервисных программ и некоторые другие встроены в Турбо Паскаль и ко всем из них есть доступ из интегрированной среды.

3.1. Вход в интегрированную среду

Для входа в интегрированную среду следует вызвать на выполнение файл ***TURBO.EXE***.

После загрузки файла ***TURBO.EXE*** на экране дисплея появится основной экран интегрированной среды, имеющий вид, показанный на рис. 22. Изображение в этом случае состоит из трех основных частей: строки основного меню, поля экрана и строки состояния. Строка основного меню содержит имена меню следующего уровня (подменю). Поле экрана предназначено для размещения открываемых окон. Строка состояния отражает состояние вычислительного процесса, а также содержит подсказки по использованию функциональных клавиш:

– **F1 – Help** (помощь);

– **F2 – Save** (записать файл);

– **F3 – Open** (вызвать файл на экран);

– **ALT – F9 – Compile** (компилировать файл);

– **F9 – Make** (компилировать файл);

– **F10 – Menu** (выход в основное меню).

Вызов определенной команды в интегрированной среде может быть осуществлен двумя методами:

– нажатием клавиш оперативного вмешательства;
– с помощью вызова функций меню (рис. 22).

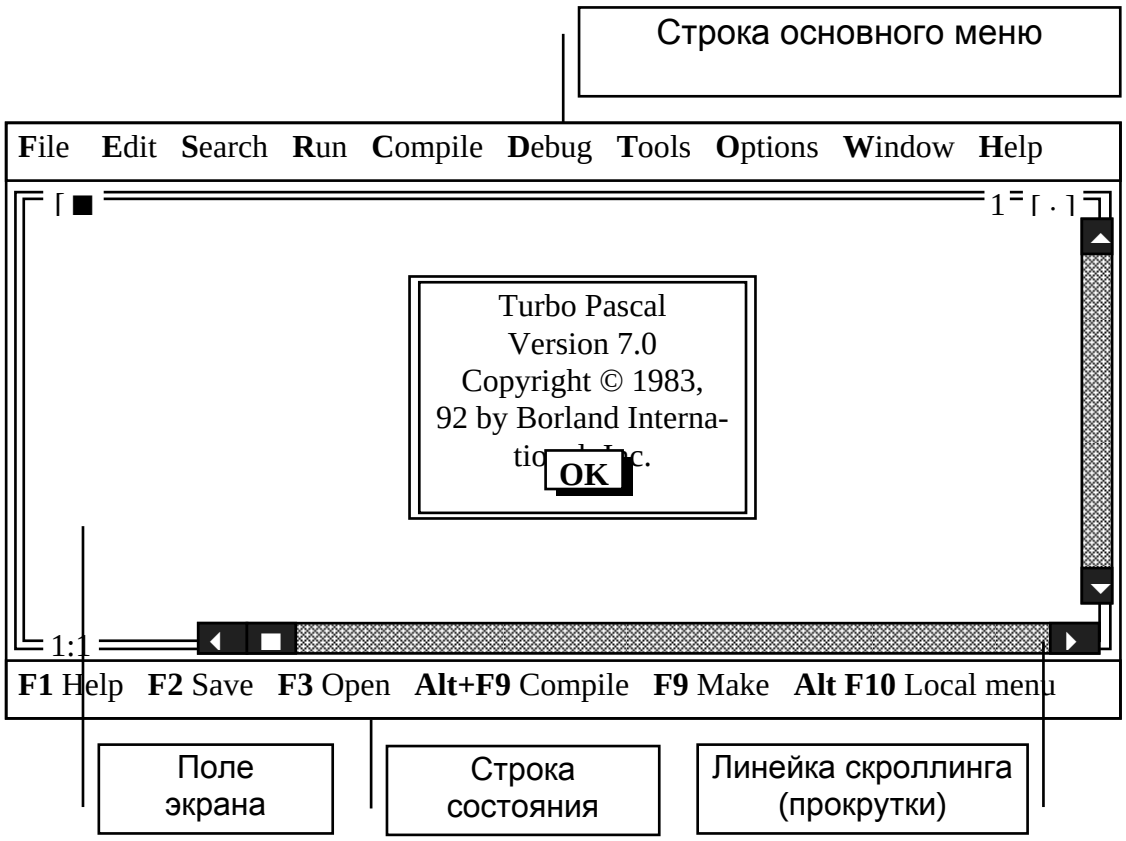


Рис. 22. Основной экран интегрированной среды

Порядок нажатия клавиш при вызове команд следующий: если названия клавиш разделены запятой (например: **F10**, **Enter)**, то их следует нажимать последовательно, если они разделены тире (например, **Ctrl – F9**), то следует нажать первую клавишу и, не отпуская ее, нажать вторую. Наиболее употребляемые клавиши оперативного вмешательства приведены в строке состояния (рис. 22). Меню Турбо Паскаля содержит набор функций. Для удобства работы однотипные функции сгруппированы в подразделы и разделы. Например, функция ***Color*** находится в разделе ***Options*** в подразделе ***Enviroment*** (условная запись ***Option*/*Enviroment*/*Color***). Для вызова функции меню требуется, во-первых, нажать **F10** для входа в меню и затем, используя **клавиши со стрелками** совместно с клавишей **Enter**, выбрать раздел, подраздел, если он есть, и функцию.

Названия разделов меню, подразделов и функций содержат символы, выделенные другим цветом. Наборные клавиши, соответствую-

щие этим символам, называются **«горячими» клавишами** (**hot keys**) и используются для ускоренного вызова функций меню. Сначала для входа в меню необходимо нажать **Alt – <«горячая» клавиша>**, после чего следует просто нажимать «горячие» клавиши. Например, для вызова функции ***Color*** необходимо нажать **Alt – O, E, C**.

3.2. Окна диалога

Некоторым функциям после их вызова необходимо указать дополнительную информацию. В этом случае на экране появится «диалоговое» окно, как, например, при нажатии **Ctrl – Q, A** (рис. 23).

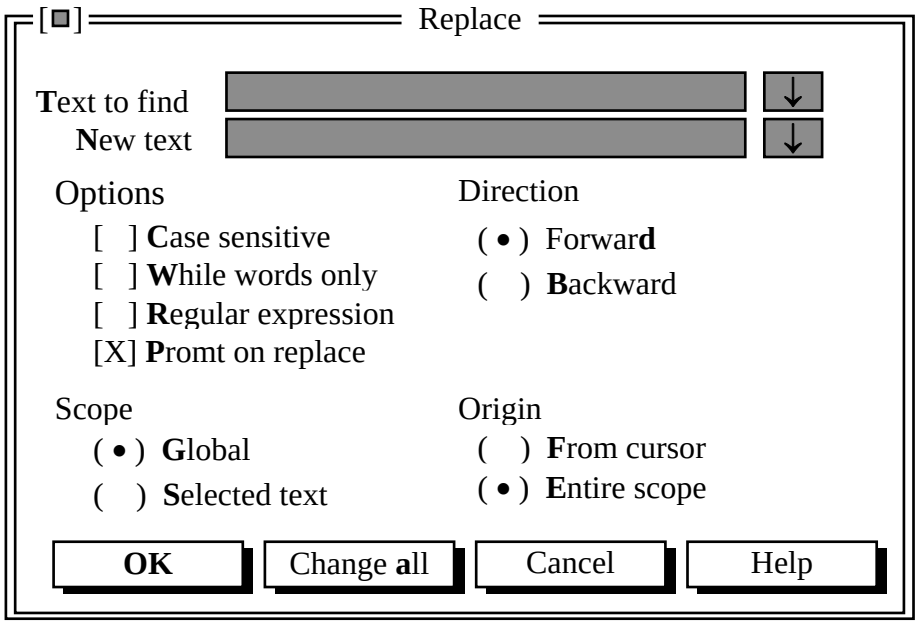


Рис. 23. Диалоговое окно

Элементами «диалогового» окна могут быть строки ввода текстовой информации, кнопки, списки выбора информации, переключатели с независимой и зависимой фиксацией. Для передвижения внутри «диалогового» окна от элемента к элементу используется **Tab** и **Shift – Tab** для передвижения в обратном направлении [3, 10].

Чтобы вызвать функцию кнопки, необходимо подвести подсветку на кнопку и нажать на **Enter**. Чтобы выбрать информацию из списка, следует, используя **клавиши со стрелками**, подвести подсветку в данную строку и нажать **Enter**. Переключатели, как правило, располагаются группами: в каждой группе несколько переключателей. Перемещение подсветки внутри группы осуществляется **клавишами со стрелками**, «включение» и «выключение» – клавишей **Space** (**пробел**).

Для передвижения используются также «горячие» клавиши (в том случае, когда нажатие клавиши приводит к вводу текста, необходимо использовать **Alt – <«горячая» клавиша>**), но при переходе на кнопку одновременно вызывается функция данной кнопки, а при переходе на переключатель он автоматически «включается» или «выключается».

Для продолжения выполнения функции меню необходимо перейти на кнопку **ОК** и нажать **Enter** или просто нажать **Enter**. Для прерывания функции меню следует перейти на кнопку **Cancel** и нажать **Enter** или нажать **Esc**.

Особенности работы с конкретными «диалоговыми» окнами будут рассмотрены ниже.

3.3. Первая программа

Этот раздел кратко описывает последовательность действий оператора-программиста при создании программы: ввести текст программы → сохранить → отредактировать → выполнить.

Чтобы начать работу на Турбо Паксале, необходимо войти в подраздел, где находятся ваши программы, нажать клавишу **F2**, запустить TURBO PASCAL. После загрузки на экране появляется основное окно компилятора Турбо Паскаля (см. рис. 22).

Создание текста программы. Для создания нового файла необходимо выполнить команду ***New*** меню ***File***. Для этого следует выполнить такую последовательность действий:

– **F10** (выход в главное меню);
– ***File*** **<Enter>**;
– ***New*** **<Enter>**

либо нажать клавиши **ALT – F, N**.

В окне редактирования появится окно со стандартным именем файла *NONAMEOO.PAS*. Далее следует набирать текст программы. При наборе можно воспользоваться командами редактора.

Сохранение текста программы. Для сохранения набранного текста программы следует выполнить команду ***Save*** меню ***File***. Для этого необходимо выполнить следующие действия:

– **F10**;
– ***File*** **<Enter>**;
– ***Save*** **<Enter>**.

Также можно воспользоваться горячей клавишей ***Save*** – **F2** (информация о «горячих» клавишах на строке подсказки). После выполнения команды ***Save*** на экране появляется окно, в котором требуется ввести имя программы, например

PROGR1.PAS <**Enter**>.

Файл с именем *PROGR1.PAS* будет сформирован в текущем каталоге. А в окне редактирования стандартное имя *NONAMEOO.PAS* будет изменено на текущее.

Выполнение. После сохранения программы можно перейти к шагу выполнения. Для этого необходимы действия:

– **F10** (выход в главное меню);
– ***Run*** **<Enter>**;
– ***Run*** **<Enter>**

либо нажать **CTRL – F9** (горячая клавиша команды ***Run***).

Перед стартом команды ***Run*** автоматически проводится компиляция программы, если она не была предварительно выполнена.

При выполнении программы возможны ошибки двух типов:

– ошибки компиляции;
– ошибки времени выполнения.

При этом курсор устанавливается в строку, приведшую к ошибке. После внесения соответствующих изменений можно вновь переходить к шагу выполнения программы.

Просмотр результатов. Просмотр результатов позволяют сделать команды:

– **F10**;
– ***Debug*** **<Enter>**;
– ***User screen*** **<Enter>**

либо команда **ALT – F5**.

После просмотра результатов для возвращения в окно Турбо Паскаля можно нажать **любую клавишу**.

Повторное обращение к программе. Для загрузки в окно редактора текста любой имеющейся программы необходимо выполнить команды:

– **F10**;
– ***File*** **<Enter>**;
– ***Open*** **<Enter>**

либо горячая клавиша **F3**.

На экране появляется список всех имеющихся программ текущего каталога. С помощью клавиш **Tab** (табуляция), ↑, ↓ **(**или манипулятора *мышь*) выбрать нужную программу и загрузить её командой **Enter**.

3.4. Главное меню

Главное меню Турбо Паскаля содержит 10 основных разделов [3, 10]: ***File***, ***Edit***, ***Search***, ***Run***, ***Compile***, ***Debug***, ***Tools***, ***Options***, ***Window***, ***Help***.

Раздел ***File*** (работа с файлами) позволяет осуществлять операции с файлами (создавать, записывать, считывать и т. п.).

Раздел ***Edit*** (редактирование) позволяет выполнять операции с фрагментами текста (копировать, вставлять, удалять фрагмент и т. п.).

Раздел ***Search*** (поиск) позволяет осуществлять поиск фрагментов текста, заменять фрагмент текста, находить ошибки и подпрограммы.

Раздел ***Run*** (выполнить) служит для компиляции и выполнения программы.

Раздел ***Compile*** (компилировать) предназначен для компиляции программы и её записи в оперативную память или на магнитный диск. Следует отметить, что на магнитный диск программа компилируется в случае создания загружаемого файла (с расширением *.EXE*) или модуля (с расширением *.TPU*).

Раздел ***Debug*** (отладка) служит для упрощения процесса отладки пользовательских программ.

Раздел ***Tools*** (инструменты) позволяет задать программы, которые можно запускать, не выходя из интегрированной среды.

Раздел ***Options*** (варианты) позволяет изменить некоторые параметры системы Турбо Паскаль, связанные с конфигурацией компьютера.

Раздел ***Window*** (работа с окнами) позволяет работать с окнами (открывать, закрывать, перемещать, изменять размеры окон и т. п.).

Раздел ***Help*** (помощь) может быть выполнен с помощью зарезервированной клавиши **F1**. Информация в системе ***Help*** снабжена комментариями, что ускоряет поиск нужной темы. Не выходя из окна редактирования, можно установить курсор на любое слово и нажать **Ctrl – F1** – на экране появится информационное сообщение.

File* – работа с файлами.** Команда ***File позволяет загружать с диска и записать на диск файлы, просматривать оглавление дисков, осуществлять временный или окончательный выход из системы Турбо Паскаль. Рассмотрим более подробно функции из подменю ***File***.

Функция ***New*** открывает окно редактора для набора новой программы или любого произвольного текста. После подачи команды на экране отображается пустое окно с наименованием *NONAMEOO.PAS*. Если программа находится на диске или дискете, её можно считать, вызвав функцию меню ***Load*** (**F3**). При этом будет открыто «диалоговое» окно для ввода имени файла (см. рис. 24) [10].

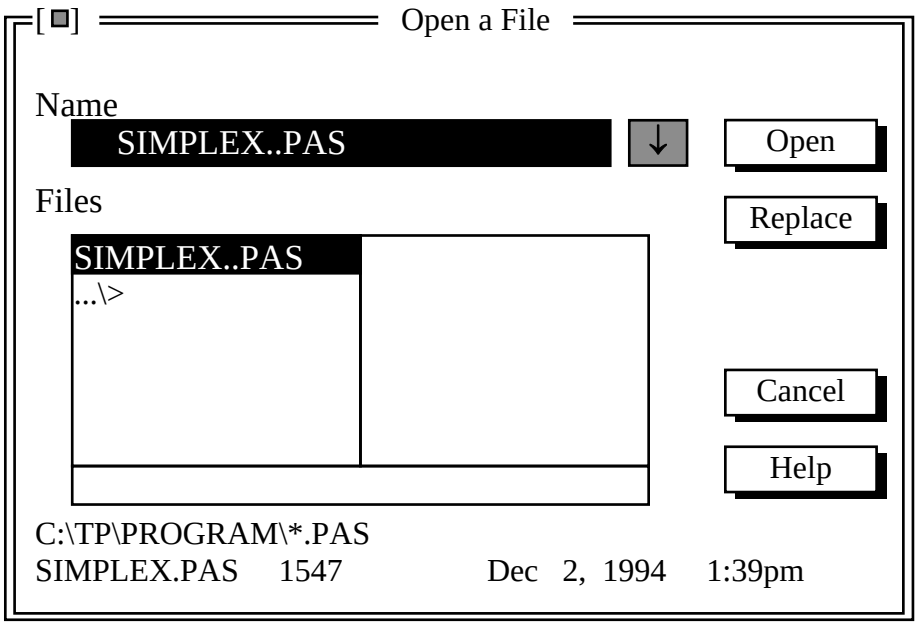


Рис. 24. Диалоговое окно для ввода имени файла

Имя файла можно ввести тремя способами:

а) набрать с клавиатуры (для программ на Паскале расширение указывать необязательно (например: *matr*, *simplex* и т. д.), для прочих текстовых файлов требуется указывать имя и расширение (например: *data.txt*, *result*. и т. д.);

б) указать формат (например: ***.pas, *.*, *.*** и т. д.) просматриваемых файлов, нажать **Enter**, выбрать имя файла из списка файлов текущего каталога;

в) нажать клавишу ↓ и выбрать имя файла из списка уже указывавшихся файлов.

Также файл можно ввести, используя список файлов, находящийся в разделе меню ***File***. Здесь указываются редактировавшиеся, но к данному моменту удаленные из редактора файлы.

Для записи программы под новым именем используется функция ***Save as***. При этом, как и в предыдущем случае, будет открыто **«диалоговое» окно** для ввода имени файла. Для записи программы под прежним именем используется функция ***Save*** (**F2**). Однако при записи только что набранных программ эта функция работает так же, как и функция ***Save as***. Команду ***Save*** рекомендуется использовать при наборе текста периодически, каждые несколько минут, во избежание потери набранного текста при сбое компьютера.

Если во время записи появится сообщение

File <имя файла> already exists. Overwrite?

Файл <имя файла> уже существует. Переписать?,

то, если находящийся на диске файл представляет собой более раннюю версию программы и он вам не нужен, нажмите – **Y**, если нет, то – **N**.

Функция ***Save all*** записывает все редактировавшиеся файлы.

Функция ***Change dir*** позволяет изменить текущий каталог. После вызова этой функции и появления **«диалогового» окна** установить каталог можно тремя способами:

– ввести с клавиатуры;

– используя «дерево каталогов» (***Dyrectory tree***) и **клавиши передвижения** совместно с **Enter**;

– используя список упоминавшихся каталогов (для вызова этого списка необходимо, находясь в ***Directory name***, нажать ↓).

Для окончания ввода нажмите **Alt – K**. Для отмены ввода – **Esc**. Для восстановления первоначального состояния дерева каталогов используется кнопка ***Revert***.

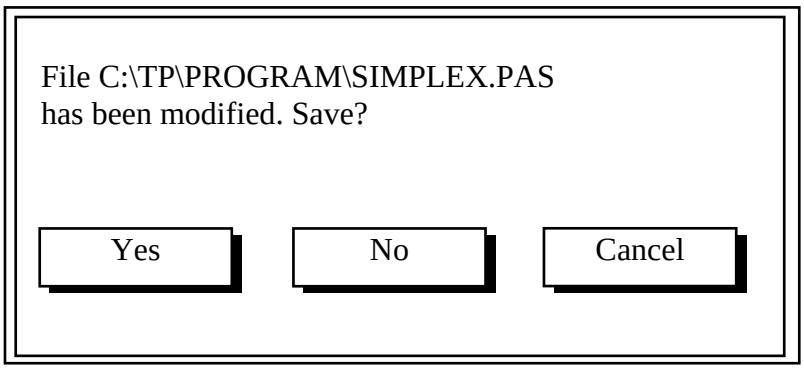
Функция ***Print*** позволяет напечатать текст программы. Функция ***Printer setup*** устанавливает параметры работы принтера.

Функция ***DOS shell*** осуществляет временный выход в DOS. Для возврата в Турбо Паскаль необходимо завершить все программы и набрать ***Exit***.

Функция ***Exit*** (**Alt – X**) – выход из Турбо Паскаля. Если, выходя из Турбо Паскаля, вы не записали редактируемые файлы, то появится сообщение:

Файл <имя файла> был изменен. Записать?

Если «да», нажмите – **Y**, если «нет» – **N**.



Работа с окнами (*Window*). ***Окно*** – это один из видимых элементов интегрированной среды Турбо Паскаля, в котором может отображаться различная информация: редактируемый текст, значения переменных и т. д. Окно имеет заголовок и может иметь порядковый номер от 1 до 9. Окно, с которым в данный момент работает программист, называется **активным** и имеет двойную рамку [3, 10].

Чтобы перейти из одного окна в другое, необходимо нажать **Alt – <номер окна>,** при этом окно с указанным номером станет активным. Эту же операцию можно осуществить, нажав **Alt – 0** (функция меню ***Window/List***), выбрать необходимый пункт и нажать **Enter**. Переходить из одного окна в другое можно также, нажимая клавишу **F6.**

Интегрированная среда запоминает порядок вашей работы с окнами. Нажав **Shift – F6** (функция меню ***Window/Previos***), вы перейдете в окно, в котором только что работали, нажав **F6** (функция меню ***Window/Next***), вы перейдете в окно, к которому дольше всего не обращались.

Можно менять размер окон. После нажатия **Ctrl – F5** (функция меню ***Window/Size/Move***) можно, нажимая **клавиши ← → ↑ ↓, Home, End, PgUp, PgDn, Shift – <← → ↑ ↓>** и затем **Enter**, изменить размер и положение окна. Клавишей **F5** (функция меню ***Window/Zoom***) можно увеличить размер окна до максимально возможного размера. Повторное нажатие **F5** уменьшает окно до первоначальных размеров.

Функция меню ***Window/Ttle*** располагает окна так, чтобы полностью было видно каждое окно. Функция меню **Window/Cascad** располагает окна друг за другом так, чтобы был виден заголовок и номер каждого окна.

Комбинация клавиш **Alt – F3** (функция меню ***Window/Close***) закрывает активное окно. Функция меню ***Window/Refresh display*** восстанавливает (перерисовывает) экран интегрированной среды.

Редактор (*Edit, Search*). Редактор – это программа, предназначенная для набора текстов и исправления ошибок в них. В данном разделе рассмотрены возможности редактора языка Турбо Паскаль.

В табл. 4 приведены функции некоторых клавиш, используемых при работе с редактором.

Таблица 4

Клавиша (и)	Выполняемая операция
1	2
Передвижение курсора	
Ctrl–S или **стрелка влево**	на символ влево
Ctrl–D или **стрелка вправо**	на символ вправо
Ctrl–A или **Ctrl–стрелка влево**	на слово влево
Ctrl–F или **Ctrl–стрелка вправо**	на слово вправо
Ctrl–E или **стрелка вверх**	на строку вверх
Ctrl–X или **стрелка вниз**	на строку вниз
Home	в начало строки
End	в конец строки

Продолжение табл. 4

1	2
Ctrl–R или **PgUp**	на страницу вверх
Ctrl–C или **PgDn**	на страницу вниз
Ctrl–Home	к верху окна
Ctrl–End	к низу окна
Ctrl–PgUp	в начало текста
Ctrl–PgDn	в конец текста
Ctrl–Q, B	в начало блока
Ctrl–Q, K	в конец блока
Del или **Ctrl–G**	удаление символа
BackSpace или **Ctrl–H**	удаление символа слева
Ctrl–Y	удаление строки
Ctrl–Q,Y; Ctrl–T	удаление до конца строки, слова справа
Восстановление	
Alt–BackSpace	восстановление текста
Работа с блоками	
Shift–<←→↑↓>	выделение блока
Ctrl–K, B	отметка начала блока
Ctrl–K, K	отметка конца блока
Ctrl–K, T	выделить слово
Ctrl–K, C	копировать блок
Ctrl–K, V	передвинуть блок
Ctrl–K, H	снятие/восстановление выделения
Ctrl–K, Y	удаление блока
Ctrl–K, W	считывание блока из файла
Ctrl–K, R	запись блока в файл
Ctrl–K, P	печать блока
Ctrl–K, I	смещение текста в блоке вправо
Ctrl–K, U	смещение текста в блоке влево
Поиск и замена текста	
Ctrl–Q, F	поиск текста
Ctrl–L	повторный поиск текста
Ctrl–Q, A	замена текста
Прочие команды	
Ctrl–W	смещение текста вверх
Ctrl–Z	смещение текста вниз
Ctrl–K, n (n = 0..9)	установка метки
Ctrl–Q, n (n = 0..9)	переход на метку
Tab или **Ctrl–I**	табуляция
Enter	раздвижение, перенос строк
Ctrl–N	раздвижение строк
Ctrl–P, Ctrl–<символ>	ввод управляющего символа
Ctrl–F1	контекстная справка
Esc	отмена команды

Окончание табл. 4

1	2
Переключение режимов редактора	
Ins или **Ctrl–V**	вставки/замещения
Ctrl–O, I	при нажатии Enter курсор переводится в следующую строку и помещается в начало строки/под первый символ в строке

Из перечисленных в таблице функций только поиск и замена требуют ввода дополнительной информации, поэтому рассмотрим их применение на примере. При нажатии **Ctrl – Q, A** на экране появляется диалоговое окно (см. рис. 23). Для выполнения операции требуется ввести заменяемый текст в строке ***Text to find***, заменяющий текст в строке ***New text***, установить, если необходимо, режимы поиска и замены:

– ***Case sensitive*** – отличие строчных и прописных букв;
– ***Whole words only*** – поиск отдельных слов;
– ***Regular expression*** – поиск математических выражений;
– ***Promt on replace*** – запрос подтверждения при замене;
– ***Forward*** – поиск сверху вниз;
– ***Backward*** – поиск снизу вверх;
– ***Global*** – поиск во всем тексте;
– ***Selected text*** – поиск в выделенном тексте (блоке);
– ***From cursor*** – поиск от курсора;
– ***Entire scope*** – поиск от начала текста.

Для однократной замены подведите подсветку на кнопку **ОК** и нажмите **Enter** или нажмите **Alt – K**. Чтобы поиск продолжался автоматически до тех пор, пока искомая строка встречается в тексте, используйте кнопку **Change all** или нажмите **Alt – A**.

Функции поиска и замены можно вызвать также, используя меню ***Search***. Здесь же содержатся функции:

– ***Go to line number*** – переход в строку с данным номером;
– ***Show last compiler error*** – показать последнюю ошибку компилятора (**Ctrl – Q, W**);
– ***Find error*** – нахождение в тексте программы ошибки, возникшей при выполнении программы;
– ***Find procedure –*** нахождение в тексте программы процедуры с данным именем.

В разделе меню ***Edit*** содержатся функции для копирования блока из одного окна в другое. Порядок копирования следующий:

а) выделить блок в одном окне;

б) вызвать функцию ***Edit/Cut*** – перенести выделенный блок в специальное окно, называемое ***ClipBoard***, или ***Edit/Copy*** – скопировать выделенный блок в ***ClipBoard***;

в) перейти в другое окно и подвести курсор к месту вставки;

г) вызвать функцию ***Edit/Paste***.

Другие функции раздела меню ***Edit***:

- ***Undo, Redo*** – восстановление удаленного текста;
- ***Clear*** – удаление блока (**Ctrl – Q, Y**);
- ***Show ClipBoard*** – показать ***ClipBoard*** в отдельном окне.

Компиляция и выполнение программы (*Compile*). Персональный компьютер PC, как и большинство микрокомпьютеров, имеет процессор, который представляет собой его рабочий механизм. Программист, как правило, составляет программу на алгоритмических языках высокого уровня, а компьютер обрабатывает программы только в машинных кодах. Для перевода программы в машинные коды служит компилятор. Компилятор Турбо Паскаля транслирует (или переводит) программу, написанную на Паскале, в команды, которые могут быть восприняты компьютером. Компилятор, таким образом, является программой, пересылающей данные: она считывает текст вашей программы и записывает его на соответствующем машинном коде.

Компиляция начинается нажатием клавиш **Alt – F9**. Если во время компиляции обнаруживается ошибка, компиляция прекращается, курсор подводится к месту ошибки и указывается номер и тип ошибки. Если при этом нажать **F1**, то можно получить информацию о данном виде ошибки на английском языке. Выход из справки – **Esc**. По завершению компиляции на экран выдается сообщение (рис. 25).

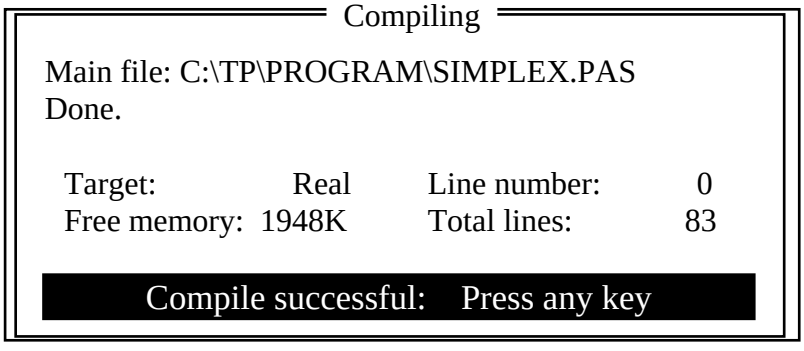


Рис. 25. Окно сообщения о завершении компиляции

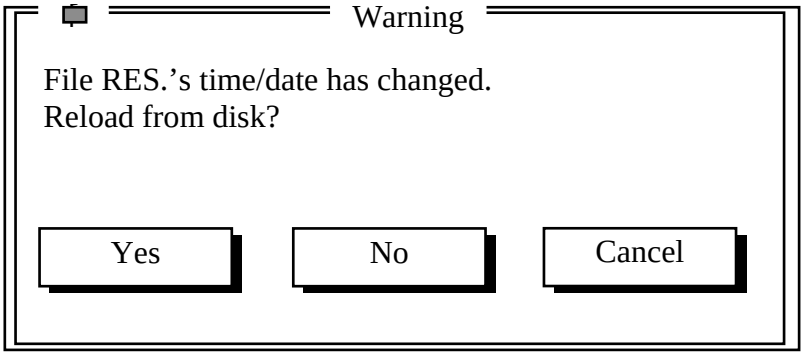
Для запуска программы необходимо нажать **Ctrl – F9**.

Если результаты расчетов вашей программы выводятся на экран, то по окончании работы программы их можно просмотреть, нажав кла-

виши **Alt – F5**. Выход из режима просмотра – **Esc**. Если результаты расчетов выводятся в файл, то по окончании работы программы этот файл можно считать с диска или дискеты. Если после этого вы вновь запустите программу, то по окончании расчетов появится сообщение

Файл был изменен. Перезагрузить?

Если «да», нажмите – **Y**, если «нет» – **N**.



Отладка программы (*Run, Debug*). В программе могут быть разнообразные ошибки: синтаксические, семантические и логические. Ошибки на этапе компиляции (или синтаксические ошибки) возникают в том случае, если написанные вами операторы Паскаля не удовлетворяют требованиям Паскаля. Другой возможный тип ошибок – это ошибки этапа выполнения (или семантические ошибки). Это происходит в том случае, если при выполнении программы предпринимается попытка выполнить недопустимое действие: открыть несуществующий файл, например, или разделить число на ноль. В этом случае Турбо Паскаль печатает на экране сообщение об ошибке, которое выглядит примерно так:

Runtime error at Seg:Ofs .

Наконец, программа может содержать логические ошибки, связанные с неправильным составлением алгоритма. Этот тип ошибок наиболее труден для обнаружения, и поэтому он может быть одной из основных причин возникновения необходимости использования отладчика.

Перед применением отладчика программа должна быть откомпилирована. Затем, нажимая **F7** или **F8**, можно начать выполнение программы «по шагам». При этом на экране появится подсветка, которая при нажатии **F7** или **F8** будет перемещаться от оператора к оператору в той же последовательности, как если бы программа выполнялась в режиме счета. Отличие клавиш **F7** и **F8** заключается в том, что при нажатии **F7** будет выполняться трассировка процедур и функций, а при **F8** вызовы процедур и функций выполняются как один шаг.

Если требуется пропустить выполнение циклов и других малоинтересных частей программы и сразу перейти к той строке, где вы хотите начать отладку, то необходимо подвести курсор в эту строку и нажать **F4**.

Точки останова. Отдельные строки программы можно пометить как точки останова. Когда вы запускаете программу, и она попадает в точку останова, ее выполнение приостанавливается, и на экран выводится текст программы в точке останова. При этом вы можете проверить значение переменных, начать трассировку (**F7, F8**) или запустить программу, пока она не достигнет следующей точки останова (**Ctrl – F9**).

Чтобы установить (или убрать) точку останова, требуется подвести курсор в требуемую строку и нажать **Ctrl – F8**. Установить точку останова можно также, вызвав функцию ***Debug/Add Breakpoint***. При этом в «диалоговом» окне требуется указать в строке ***Condition*** – логическое выражение – условие остановки (используется при отладке циклов ***Repeat Until*** и ***While Do***), ***Path count*** – число, сколько раз будет пропущена точка останова перед остановкой на ней (используется при отладке цикла ***For***), ***File name*** – имя файла, в котором поставлена строка останова, ***Line number*** – номер строки. Работать с уже установленными точками останова можно, используя функцию меню ***Debug/Breakpoints***. При этом в появившемся «диалоговом» окне, используя клавиши со стрелками, можно выбрать необходимую точку останова и, вызывая функции кнопок:

- изменить параметры точки останова (кнопка **Edit**);
- удалить точку останова (кнопка **Delete** или клавиша **Del**);
- перейти на выбранную точку останова (кнопка **View**);
- удалить все точки останова (кнопка **Clear all**).

В добавление к использованию в программе точек останова работающую программу в любой момент можно приостановить либо остановить, нажав **Ctrl – Break**. Как правило, однократное нажатие **Ctrl – Break** приводит к приостановке программы с возможностью дальнейшей отладки, а двукратное – к остановке.

Окно просмотра. Турбо Паскаль дает программисту возможность во время отладки просматривать и изменять текущие значения переменных. Для просмотра переменных служит окно просмотра.

Окно просмотра можно открыть двумя способами: вызовом функции меню ***Debug/Watch*** или первым добавлением переменной, структуры данных или выражения в окно просмотра. При этом окно просмотра помещается в нижней части экрана, и его размеры и положение на экране можно изменить, используя клавиши работы с окнами.

Добавление переменных, структур данных и выражений может быть осуществлено нажатием **Ctrl – F7** (или вызовом функции меню ***Debug/Add watch***).

При этом появится «диалоговое» окно. Необходимый текст можно ввести с клавиатуры или, нажав клавишу ↓, выбрать его из списка. Также для ускорения добавления можно подвести курсор на необходимую переменную в редакторе, нажать **Ctrl – F7** и, если не весь требуемый текст появился в окне, нажимая →, добиться появления всего текста.

Добавленные переменные, структуры данных и выражения могут быть изменены или удалены. Для этого необходимо войти в окно просмотра, выбрать, используя клавиши ← → ↑↓, требуемую строку и для изменения нажать на **Enter** или для удаления – на **Del**.

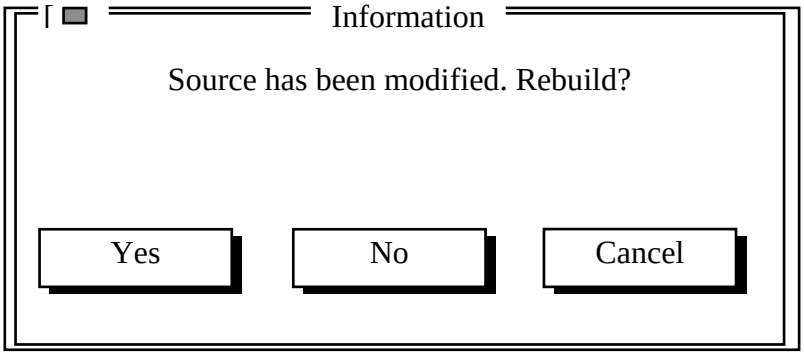
Кроме окна просмотра в отладчике можно использовать специальные окна.

Окно «стек вызова» (***Debug/Call Stack*** или **Ctrl – F3**). При каждом вызове процедуры или функции Турбо Паскаль запоминает вызов и передаваемые ей параметры. Когда вы выходите из этой процедуры или функции, то Паскаль про этот вызов «забывает». Это называется стеком вызова. Используя окно «стек вызова», можно просмотреть список вызовов процедур и функций и переданные им фактические параметры. Окно «стек вызова» несет также другую важную функцию: оно позволяет выполнять обратный просмотр последовательности вызовов. Когда на экране появится стек вызова, самый верхний вызов подсвечивается. Для перемещения вверх и вниз по стеку можно использовать клавиши передвижения курсора. Если вы нажмете клавишу **Enter**, то переместитесь к последней активной точке в программе или подпрограмме.

Окно просмотра (функция ***Debug/Output***) позволяет просматривать всю информацию, выводимую программой на экран.

Завершение сеанса отладки осуществляется нажатием **Ctrl – F2** (функция ***Run/Program Reset***). С другой стороны, если вы во время отладки модифицировали какую-либо часть программы, то при нажатии одной из клавиш выполнения (**F7, F8, Ctrl – F9** и т. д.) вы получите сообщение

Source modified, rebuild? (*Y/N*)
Исходный код изменен, сформировать заново?



Если вы нажмете **Y** (да), то Турбо Паскаль выполнит повторное создание вашей программы и начнет отладку сначала. Если вы нажмете **N** (нет), то Турбо Паскаль, предполагая, что вы знаете, что делаете, продолжит сеанс отладки (изменения не будут учтены).

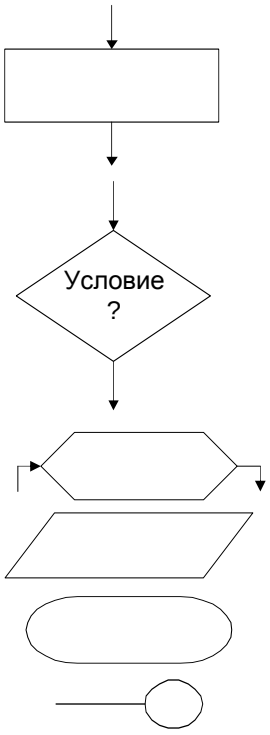
4. ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ТУРБО ПАСКАЛЬ

4.1. Алгоритмы

Практически решение любой задачи на ЭВМ начинают с составления алгоритма.

Алгоритм – это точно определенная последовательность элементарных действий, необходимых для решения задачи. Алгоритм содержит несколько шагов, которые должны выполняться в определенной последовательности. Каждый шаг алгоритма может состоять из одной или нескольких простых операций [2, 13, 17].

Наиболее наглядным способом описания алгоритмов является описание его в виде блок-хем. При этом алгоритм представляется последовательностью блоков, выполняющих определенные функции, и связей между ними. Внутри блоков указывается информация, которая характеризует, какие функции выполняют данные блоки. На блок-схеме каждый шаг алгоритма обозначается специальной геометрической фигурой, а внутри записываются простые операции. Для изображения блок-схем программ используются следующие наиболее часто встречающиеся обозначения.



Выполнение операции или групп операций

Проверка условия

Выполнение циклических операций

Ввод исходных данных и вывод результатов

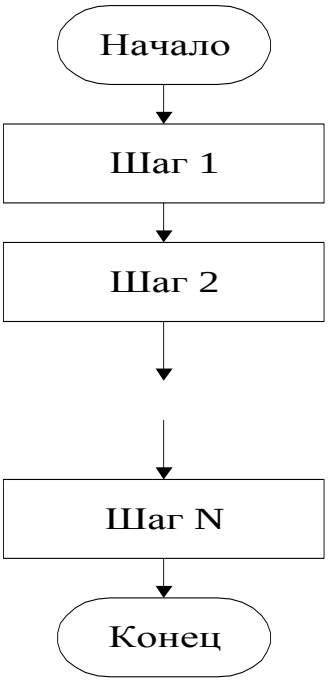
Начало, конец

Указание связи между прерванными линиями, связывающими блоки

Существует несколько типов алгоритмов.

4.1.1. Линейный алгоритм

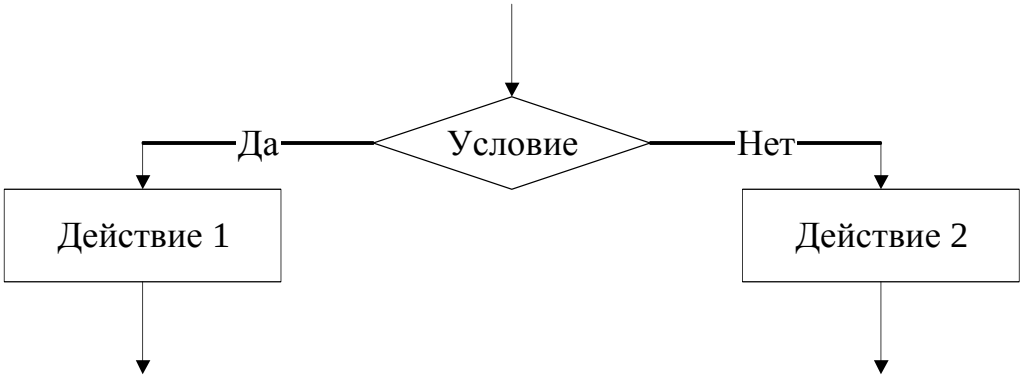
Линейным называется алгоритм, который содержит N шагов и все шаги выполняются последовательно друг за другом без разветвлений.



4.1.2. Разветвляющийся алгоритм

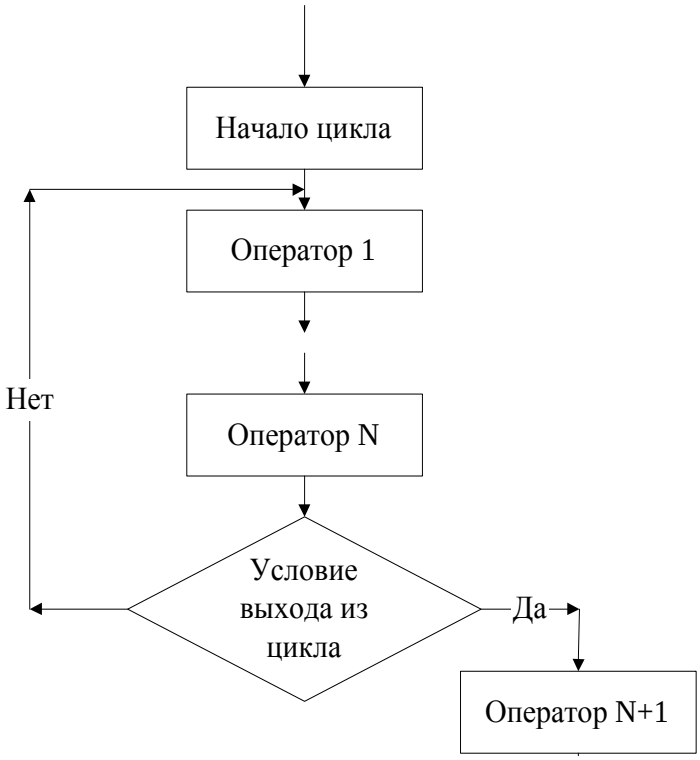
Разветвляющимся называется такой алгоритм, в котором последовательность выполнения операций (шагов) изменяется в зависимости от некоторых условий и продолжается в различных направлениях.

Если условие верно, то выполняется действие 1, в противном случае – действие 2.



4.1.3. Циклический алгоритм

Циклическим называется такой алгоритм, который содержит многократно повторяющийся участок в зависимости от заданной величины (параметра цикла). Ниже приведена блок-схема циклического алгоритма с проверкой условия для параметра в конце цикла.



4.2. Введение в Турбо Паскаль

Любая ЭВМ обрабатывает информацию по программам.

Программой называется алгоритм, записанный на языке программирования. Существуют различные языки программирования, и каждый из них предназначен для решения определенного круга задач. Для решения вычислительных задач используются операторные языки программирования. К таким языкам относятся FORTRAN, PASCAL, BASIC и некоторые другие. С помощью этих языков удобно решать математические, физические, инженерные задачи. Но программы для решения нечисловых задач лучше писать на других языках [1, 11].

В операторных языках каждый элемент алгоритма записывается с помощью какого-либо оператора. Каждый оператор выполняет определенное действие (определенную операцию) в программе.

Алгоритмический язык Паскаль появился в 1970 г. Автор языка Паскаль – Никлаус Вирт (профессор, директор института Информатики в Швейцарии). Язык назван в честь французского ученого Блеза Паскаля, разработавшего одно из первых суммирующих устройств.

Язык программирования Паскаль является достаточно простым и компактным языком, который широко применяется в мини-ЭВМ и ПЭВМ. Конструкции языка позволяют работать не только со стандартными типами данных, но и представляют пользователю возможность выполнять операции над файлами, множествами и записями, использовать динамические структуры данных [10–17].

4.2.1. Символы, простейшие конструкции языка

В языке Турбо Паскаль используются следующие символы.

Буквы: 26 латинских букв (прописных и строчных).

Цифры: 0, 1, ..., 9.

Специальные символы:

: + - * / < > = ( ) [ ] / , ; _ { }

Ключевые слова: **AND** – и, **ARRAY** – массив, **BEGIN** – начало, **CASE** – вариант, **CONST** – константа, **DIV** – деление нацело, **DO** – выполнять, **DOWNTO** – уменьшать до, **END** – конец, **FILE** – файл, **FOR** – для, **FUNCTION** – функция, **GOTO** – перейти на, **IF** – если, **IN** – включение, **LABEL** – метка, **MOD** – модуль, **NIL** – отсутствие указателя, **NOT** – не, **OF** – из, **OR** – или, **PACKED** – упакованный, **PROCEDURE** – процедура, **PROGRAM** – программа, **RECORD** – запись, **REPEAT** – повторять, **THEN** – то, **TO** – до, **TYPE** – тип, **VAR** – переменная, **UNTIL** – до, **WHILE** – пока, **WITH** – с.

Знаки операций:

– арифметических: + (сложение), – (вычитание), * (умножение), / (деление), DIV (деление нацело с отбрасыванием остатка), MOD (нахождение остатка от деления нацело);

– отношения: > (больше), < (меньше), <= (меньше или равно), >= (больше или равно), = (равно), <> (не равно);

– логических: NOT (отрицание), OR (логическое сложение), AND (логическое умножение).

Имена (идентификаторы). Для обозначения различных объектов в языке Паскаль используются их имена или идентификаторы.

Имя (идентификатор) в Паскале – это произвольная последовательность букв и цифр, начинающаяся с буквы. Русские буквы использовать нельзя. Например:

A X12 UM1 TY134B.

Если необходимо разделить имя, можно использовать символ подчеркивания:

DATE_27_sep A_B.

Длина идентификатора может быть любой, но значимыми являются только первые 63 символа.

Константы. Константами называются параметры программы, значения которых не меняются в процессе её выполнения.

В Турбо Паскале используются три вида констант: числовые, символьные и строковые, логические.

1. ***Числовые*** константы. В Паскале используются целые и вещественные (действительные) числа. *Целые* числа имеют обычный арифметический вид (знак, цифры):

0, 1, +100, –12.

Диапазон изменения целых чисел зависит от типа ЭВМ.

Вещественные числа имеют две формы представления: в виде обычной десятичной дроби с фиксированной точкой:

2.75, –11.2

и числа с плавающей точкой, представленные в форме: mEp, где m – мантисса, E – признак записи числа, p – порядок числа;

$1.2 \cdot 10^{-5}$ соответствует 1.2E–5.

Представление чисел

целое	с фиксированной точкой	с плавающей точкой
–257	–257.0	–2.57E2
3	3.0	3.0E0

Константы с фиксированной точкой обязательно должны содержать как целую, так и дробную часть: 2.0; 0.5.

2. ***Символьные*** константы и константы-строки.

Символьная константа – это символ, заключенный в апостроф: 'A', '!', '+', ‘8’.

Строковая константа – это последовательность символов, заключенных в апострофы:

'MASSA', 'СКОРОСТЬ', '2МПа'.

3. ***Логические*** константы – принимают два значения: True (истина) и False (ложь).

Переменные. Переменными называются параметры программы, значения которых могут изменяться в процессе ее выполнения. Переменная в Паскале имеет имя и тип.

Тип:

– real – вещественный 1.5;
– integer – целый 5;
– boolean – логический true, false;
– char – литерный 'a'.

Различают простые переменные и переменные с индексом, в записи которых указывается имя массива и индекс этого элемента.

Например: VOL, X, A, B[1], P[1,3], C[i].

Комментарии. Служат для пояснений в программе. Комментарии – это любой текст, заключенный в { } или (* *). Например:

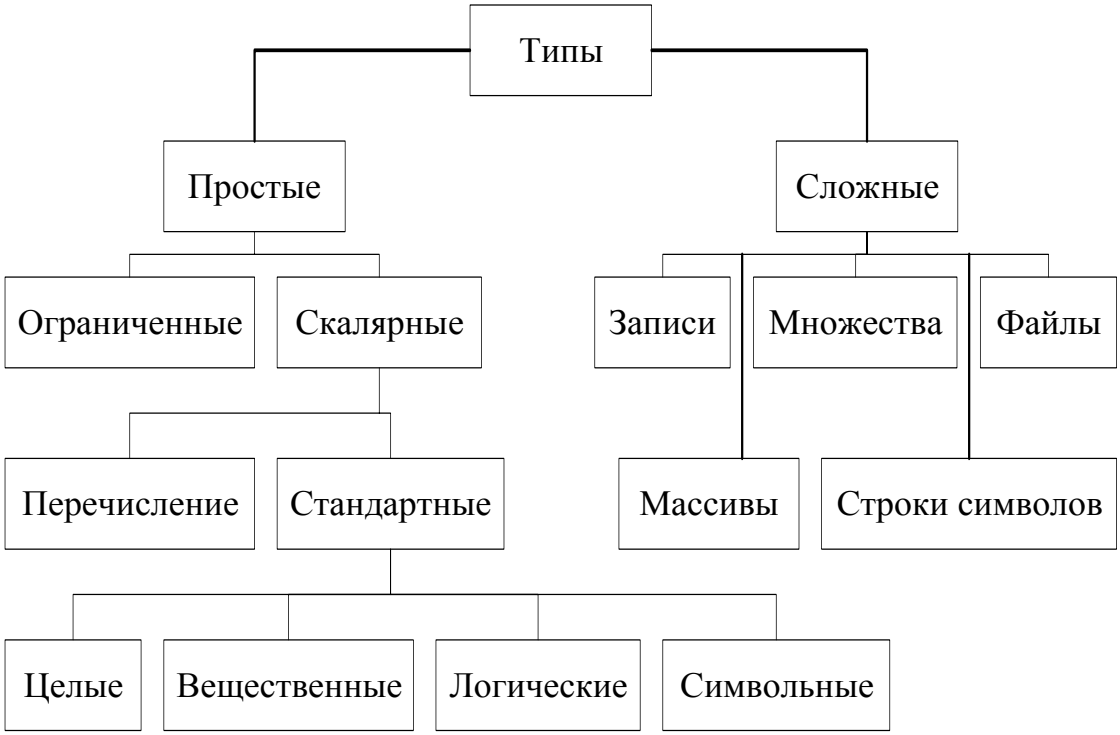
{ Расчет суммы} или (*Расчет суммы *).

Комментарии можно помещать в любом месте программы, они не воспринимаются и не обрабатываются ЭВМ.

4.2.2. Типы данных

В языке Турбо Паскаль под типом понимается множество значений, которые может принимать переменная, а также совокупность операций, которые можно выполнять с этими значениями.

В Турбо Паскале можно выделить группы простых и сложных типов. Основные типы переменных в Турбо Паскале можно представить в виде следующей схемы:



Среди типов, используемых в языке, есть стандартные (предопределенные) и определяемые программистом.

К стандартным типам относятся целый, вещественный, логический и символьный типы. Все другие используемые типы данных (нестандартные) должны быть определены в разделе объявления типов, который начинается словом **type**, за которым следует имя типа и список его значений:

type <имя типа>=<определение типа>.

Целый тип. Обеспечивает задание целых чисел. В Турбо Паскале имеется пять стандартных типов целых чисел. Характеристики этих типов приведены в табл. 5.

Таблица 5

Целые типы данных

Тип	Диапазон	Формат	Размер в байтах
Shortint	–128 ÷ 127	Знаковый	1
Integer	–32768 ÷ 32767	Знаковый	2
Longint	–2147483648 ÷ 147483647	Знаковый	4
Byte	0 ÷ 255	Беззнаковый	1
Word	0 ÷ 65535	Беззнаковый	2

Над переменными целого типа определены следующие арифметические операции: сложение (+), вычитание (–), умножение (*), деление (/); div – деление нацело, mod – вычисление остатка от целочисленного значения. Результат выполнения этих операций будет целого типа, кроме операции деления, результат которой всегда вещественного типа.

Например: 7 div 2 = 3; 7 mod 3 = 1; (–7) div 2 = –3;
14 mod 3 = 2; 3 div 5 = 0; 8/4 = 2.0; 12/3=4.0.

Вещественный тип. В отличие от стандартного языка Паскаль, где определен только один вещественный тип – real, в Турбо Паскале имеется пять стандартных вещественных типов (см. табл. 6).

Таблица 6

Вещественные типы данных

Тип	Диапазон	Формат	Размер в байтах
Real	2.9E–39 ÷ 1.7E+38	11–12	6
Single	1.5E–45 ÷ 3.4E+38	7–8	4
Double	5.0E–324 ÷ 1.7E+308	15–16	8
Extended	3.4E–4932 ÷ 1.1E+4932	9–20	10
Comp	–9E18 ÷ 9E18	19–20	8

Типы Single, Double, Extended и Comp можно использовать в программах только при наличии арифметического сопроцессора.

Символьный тип. Стандартный символьный тип char определяется множеством значений кодовой таблицы ПЭВМ. Все символы клавиатуры упорядочены (пронумерованы) и каждая литера имеет свой порядковый номер. Над символьными значениями не предусмотрены никакие операции, но они могут сравниваться, участвовать в чтении, печати и в операциях присваивания. Например:

```
var C1,C2: char;
 . . . . .
C1:='A';
C2:='Z';
if C1<C2 then C1:='Z' else C2:='A';
. . . .
```

Логический тип. Стандартный логический тип Boolean (размер – 1 байт) – это тип данных, любой элемент которого может принимать только два значения: True и False.

Над логическими переменными можно осуществлять логические операции:

– and (и) – логическое умножение;
– or (или) – логическое сложение;
– not (не) – логическое отрицание.

Перечисляемый тип. Этот тип задается перечислением всех значений, которые может принимать переменная данного типа. Описание этих переменных имеет вид:

type <имя типа> = (список значений);
var <имя переменной> : <имя типа>;

или

var <имя переменной> : <список значений>;

Например:

```
type God=(Zima, Vesna, Leto, Osen);
var A1,A2:God;
```

или

```
var A1,A2: (Zima, Vesna, Leto, Osen);
```

Здесь A1 и A2 переменные типа «перечисление», которые могут принимать любые из заданных значений.

Для переменных перечисляемого типа применимы операции отношения: <, >, >=, <=, =, если оба компонента отношения имеют один тип. Описание типа переменной одновременно упорядочивает ее значения, так: Zima < Vesna < Leto < Osen.

Ограниченный тип (тип-диапазон). Этот тип позволяет задавать две константы, определяющие границы диапазона значений для данной переменной. Описание имеет следующий вид:

type <имя типа> = Min .. Max;
var <имя переменной> : <имя типа>;

или

var <имя переменной> : Min .. Max;

Например:

```
var P:1..10
```

Здесь Min и Max – соответственно константы, определяющие левую и правую границы значений, которые может принимать ограниченная переменная. Обе константы должны принадлежать одному из стандартных типов, кроме real. Обязательно: Min < Max.

К значениям ограниченного типа применимы все операции и функции, которые определены над значениями базового типа.

4.2.3. Структура программы

Общую структуру программы на Турбо Паскале для простых алгоритмов можно представить:

Program <имя>;
  <раздел описаний>
  label <список меток>;
  const <список констант>;
  type <список типов>;
  var <список переменных>;
  procedure, function <список процедур, функций>;
Begin
  <тело программы>
End.

Раздел описаний:

1. Раздел меток (label).

```
label 5, 10, M1, M2; { список меток }.
```

Метка может содержать цифры от 0 до 9999, но не может иметь больше четырех знаков.

Например:

```
10: A:=A+1.
```

2. Раздел констант (const).

```
const a=2.56; R=1.198; P=1.75E+2;
      f:real=-0.5;
```

Изменять значения простых констант в программе нельзя.

3. Раздел типов.

В этом разделе описываются имена типов переменных, отличные от стандартных. Например, массивы:

```
type mas= array [1..5] of real;
```

4. Раздел описания переменных.

Каждая переменная в программе должна быть описана в разделе описания переменных:

var <переменная>:<тип;

Например:

```
var a,cl,SK:real;
    P1,P2: char;
    P: array[1.5] of real; {описание массива}
    b: boolean;
    a,x: integer;
```

5. Раздел операторов.

Тело программы начинается словом **Begin** и заканчивается словом **End** с точкой, которая является признаком конца программы. Раздел операторов – выполняемая часть программы, которая записывается в свободной форме. Операторы отделяются друг от друга точкой с запятой. Допускается располагать несколько операторов в одной строке, а также переносить с одной строки на другую части описаний или операторов (но без разделения ключевых слов и идентификаторов). Пробелы допускаются в любом месте программы и в неограниченном количестве. Между ключевыми словами обязателен пробел. Внутри ключевого слова пробел не допускается.

4.2.4. Стандартные функции

Стандартные функции используются для вычисления часто встречающихся функций. При обращении к стандартным функциям необходимо записать имя функции, а в скобках указать аргумент. Основные стандартные функции и процедуры приведены в табл. 7.

Таблица 7

Основные стандартные функции

Функция	Назначение	Тип	
		аргумента	функции
abs(x) sqr(x)	$\|x\|$ x^2	real или integer	real или integer
sin(x) cos(x) exp(x) ln(x) sqrt(x) arctan(x)	sin x cos x e^x ln x $\sqrt{x}$ arctg x	real или integer	real
trunc(x) int(x) frac(x) round(x)	Вычисление целой части числа x Вычисление целой части числа x Вычисление дробной части числа x Округление числа x в сторону ближайшего целого	real real real real	integer real real integer
pred(x) succ(x)	Нахождение предшествующего элемента Нахождение последующего элемента	integer или char или boolean	integer или char или boolean
ord(x)	Определение порядкового номера символа в наборе символов	char или boolean	integer
chr(i)	Определение символа из набора символов по номеру i	integer	char
inc(x)	Увеличение значения x на единицу		
dec(x)	Уменьшение значения x на единицу		
odd(x)	Определение четности числа: true, если x нечетное false, если x четное	integer	boolean

Примеры использования некоторых стандартных функций:

Функция	Результат	Функция	Результат
Trunc(6.3)	6	Int(7.3)	7.0
Trunc(6.7)	6	Int(7.8)	7.0
Round(8.3)	8	Int(-1.2)	-2.0
Round(8.9)	9	Frac(9.3)	0.3
Round(-3.5)	-4		

В Турбо Паскале **нет операции возведения в степень**, ее заменяют выполнением следующей операции:

$$x^a = \exp(a \cdot \ln(x)); \qquad \sqrt[a]{x} = x^{1/a} = \exp(1/a \cdot \ln(x)).$$

Вычисление логарифмов производят по соотношениям:

$$\log_a x = \ln x / \ln a; \quad \lg x = \ln x / \ln 10.$$

В Паскале определены только три тригонометрические функции: sin, cos, arctg. Для вычисления остальных тригонометрических функций необходимо использовать соотношения:

$\operatorname{tg} x = \sin x / \cos x;$

$\operatorname{ctg} x = \cos x / \sin x;$

$\operatorname{csc} x = 1 / \sin x;$

$\operatorname{sc} x = 1 / \cos x;$

$\arcsin x = \operatorname{arctg}(x/(1-x^2))^{1/2};$

$\arccos x = \pi/2 - \arcsin x;$

$\operatorname{arcctg} x = \pi/2 - \operatorname{arctg} x.$

4.2.5. Выражения

Выражение – это синтаксическая единица языка, определяющая способ вычисления некоторого значения. Выражения в языке Паскаль формируются в соответствии с определенными правилами из констант, переменных, функций, знаков операций и круглых скобок [10–16].

Начинается вычисление с определения переменных и констант, входящих в выражение. Дальнейшие действия выполняются в соответствии с их приоритетом. В первую очередь вычисляются выражения, заключенные в круглые скобки, далее – значения входящих в выражение функций и т. д. Операции одного приоритета выполняются последовательно слева направо.

При вычислении выражений принят следующий приоритет операций:

- арифметических:
  - вычисление значений стандартных функций;
  - умножение и деление;
  - сложение и вычитание;

- логических:
  - not;
  - *, /, div, mod, and;

– +, –, or;

♦ отношения:

– <=, >=, <, >, =.

Тип результата выражения зависит от типов операндов, участвующих в операции. Тип результата операций «+», «*», «–» является INTEGER, если оба операнда имеют тип INTEGER, и REAL – в противном случае. Результатом операции «/» всегда является тип REAL. Результат выполнения логических операций NOT, OR, AND всегда имеет тип BOOLEAN. Аргументы операций сравнения на равенство и неравенство (=, < >) могут иметь любой тип переменных и констант, а результат всегда имеет тип BOOLEAN. В операциях сравнения (>, <, >=, <=) аргументы могут быть любого типа, а результат имеет только тип BOOLEAN.

Примеры записи выражений:

$a \cdot e^{2t} - \sqrt{x \cdot y \cdot z}$;	a*exp(2*t)–sqrt(x*y*z);
$a \cdot x^2 + (4 \cdot a \cdot b - x \cdot c)/2$;	a*x*x+(4.0*a*b–x*c)/2);
$\mathrm{tg}^2 x + b^3 - \lg b^2 \cdot \sqrt[3]{\sin x}$	sqr(sin(x)/cos(x))+exp(3*ln(b))– ln(b*b)/ln(10)*exp(1/3*ln(sin(x))).

4.3. Операторы языка

Операторы языка описывают некоторые алгоритмические действия, которые необходимо выполнить для решения задачи. Тело программы можно представить как последовательность таких операторов. Операторы программы разделяются точкой с запятой. Все операторы языка Паскаль можно разбить на две группы: простые и структурированные [10–17].

4.3.1. Простые операторы

Простыми являются те операторы, которые не содержат других операторов. К ним относятся:

– оператор присваивания;
– оператор безусловного перехода GOTO;
– пустой оператор.

4.3.1.1. Оператор присваивания

С помощью этого оператора переменной присваивается значение выражения. Для этого используется знак присваивания **«:=».** Общий вид оператора следующий:

<имя переменной>:= выражение.

В операторе присваивания переменная и выражение должны иметь один и тот же тип. Однако допускается присваивать переменной типа real выражение типа integer. Присваивание же переменной целого типа выражения вещественного типа запрещается.

Пример. Вычислить значение концентрации вещества по формуле $C=P/RT$, при P = 10 ат ; T = 513 К; R= 0,001986 ккал/моль·К.

```
Program Conc;
 const R=1.986E-3;
 var P,T:integer;
     C: real;
begin
 P:=10;
 T:=513;
 C:=P/(R*T);
 writeln('C=',C);
end.
```

4.3.1.2. Оператор безусловного перехода GOTO

Оператор GOTO позволяет изменить стандартный последовательный порядок выполнения операторов в программе и перейти к выполнению программы, начиная с заданного оператора. Общий вид оператора следующий:

goto n,

где n – метка оператора.

Метки, используемые в Турбо Паскале, могут быть двух типов:

- целым числом в пределах от 0 до 9999;
- обычным идентификатором.

Метка должна быть описана в разделе **label**. Одной меткой можно пометить только один оператор. Например:

```
goto 20;
10: B:=3;
. . . .
20: X:=X/B;
goto 10;
```

4.3.1.3. Пустой оператор

Пустой оператор – это оператор, не выполняющий никакого действия. Он используется для выхода из середины программы или составного оператора. Пустому оператору соответствует символ «;». Чаще всего пустой оператор встречается с меткой. Например:

```
goto 10;
. . .
10: ;
End.
```

Символ «;» можно опустить, тогда можно записать

```
10: End.
```

4.3.1.4. Ввод-вывод данных

Для ввода и вывода данных в Турбо Паскале существуют стандартные процедуры ввода-вывода, вызываемые соответственно операторами READ и WRITE [10–16].

Операторы ввода:

1) Read (<список переменных>) – последовательный ввод переменных из списка;

2) Readln (<список переменных>) – то же, что и оператор Read, только после ввода данных происходит переход на новую строку, т. е. ввод осуществляется каждый раз с новой строки;

3) Readln – происходит переход на новую строку без ввода данных.

Значения вводимых переменных должны соответствовать типам переменных из списка ввода. В Турбо Паскале допускается вводить значения следующих данных: целых (integer), вещественных (real), символьных (char), строковых (string).

С помощью оператора ввода нельзя ввести:

1) значение логической переменной;

2) значение переменной типа «массив» (необходимо вводить значения отдельных элементов массива);

3) значение переменной типа «перечисление»;

4) значение переменной типа «запись».

Ввод в языке Паскаль может быть только бесформатный. Данные набираются на терминале, разделителем между числами служит пробел. Разделитель между символами, между числом и символом не нужен.

Пример 1. Требуется ввести значения следующих переменных:

A=2.5; S=7.42; P= –0.34E–01; M=10; N=15; C1='B'; C2='K'; C3='E'

```
var A,S,P:real;
    M,N:integer;
    C1,C2,C3:char;
 . . . . . .
    Readln(A,S,P);
    Read(M,N,C1,C2,C3);
```

Значения переменных вводятся следующим образом:

```
 2.5 7.42 -0.34E-01
10 15BKE (без апострофов)
```

или

```
2.5
7.42
-0.34E-01
10 20BKE
```

Операторы вывода. Оператор вывода данных имеет три формы записи:

1) Write (<список переменных>) – выводит последовательно значения переменных из списка;

2) Writeln (<список переменных>) – то же, что и оператор Write, но после вывода переменных осуществляется переход на новую строку (следующий оператор вывода будет выводить данные с начала новой строки);

3) Writeln – осуществляет переход на новую строку без вывода данных.

В Турбо Паскале допустим вывод значений данных следующих типов: целых, вещественных, символьных, логических и строковых.

Значения величин действительного типа выводятся в нормализованном виде (мантисса числа с порядком), а целого типа – в обычной форме.

Пример 2. Пусть в результате выполнения программы переменные получили следующие значения:

K= –7; P=8.74; S='+'; C=True.

Выведем их на печать:

```
  var K: integer; P: real; S: char; C: boolean;
Begin
```

```
  . . . . . .
  Writeln ('  Пример');
  Writeln ('K=',K,' P=',P);
  Writeln ('S=',S);
  Write ('C=',C);
End.
```

Информация будет выведена в следующем виде:

```
  Пример
K=-7 P=8.7400000000E+0000
S=+
C=True
```

В Турбо Паскале предусмотрен вывод данных по формату. Общий вид записи для вывода значений *целого* типа:

Write (p:m),

где p – имя переменной, m – число позиций, отводимое для выводимой величины p.

Для вывода значений *действительного* типа:

1) **Write (p:m); 2) Write (p:m:n),**

где m – общее число позиций для выводимой переменной p, включая знак числа, целую часть, точку и дробную часть; n – число позиций дробной части.

Оператором (1) переменная вещественного типа p выводится в виде константы с плавающей точкой с шириной поля m.

Оператором (2) переменная p выводится в виде константы с фиксированной точкой.

Пример 3. Используем форматный вывод для следующих переменных:

K= –7; P=8.74; X=3.524; X1= $0.264*10^{-5}$; S='+'; C=True.

```
Writeln ('K=',K:3,' P=',P:5:2);
Writeln ('X=',X:10,' X1=',X1:10);
Writeln ('S=',S:2,',',' C=',C:6);
```

В результате вывода получим

```
K= -7 P= 8.74
X= 3.5240E+00 X1=0.2640E-05;
S= +, C= True
```

4.3.1.5. Программирование линейных алгоритмов

Рассмотрим пример программы для линейного алгоритма.

Пример. Вычислить скорость химической реакции по выражению

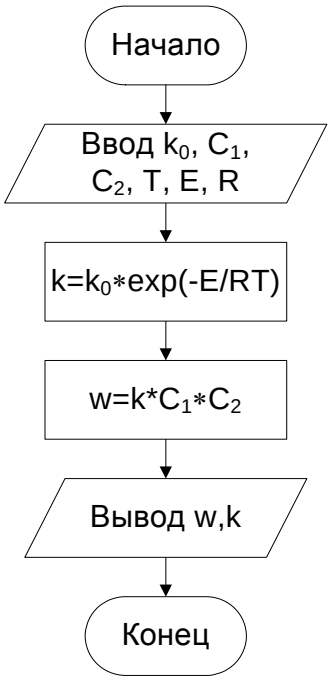
$$w = k \cdot C_1 \cdot C_2,$$

в котором константа скорости рассчитывается по формуле Аррениуса

$$k = k_0 \cdot e^{-E/RT},$$

где $k_0 = 0.34 \cdot 10^7$; $C_1 = 1.2$ моль/л, $C_2 = 0.2$ моль/л; $E = 22.1$ ккал/моль; $T = 473$ К; $R = 0.00198$ ккал/моль·К.

Блок-схема алгоритма имеет следующий вид.




```
Program Scor;
 var W,k0,k,C1,C2,E:real;
     T:integer;
 const R=0.198E-2;
Begin
 K0:=0.34E7; E:=22.1; C1:=1.2; C2:=0.2;
 writeln('введи T=');
 readln(T);
 k:=k0*exp(-E/(R*T));
 W:=k*C1*C2;
 writeln('k=',k,' W=',W);
End.
```

4.3.2. Структурированные операторы Паскаля

Структурированными являются такие операторы, которые состоят из других операторов [10–16]. К ним относятся:

– составной оператор;
– условный оператор IF;
– условный оператор CASE;
– операторы цикла WHILE, REPEAT, FOR.

4.3.2.1. Составной оператор

Составной оператор позволяет объединить несколько операторов Паскаля в одну конструкцию, которая рассматривается как составной оператор. Общий вид оператора следующий:

```
begin
 оператор 1;
 оператор 2;
  . . . . . .
 оператор n
end;
```

В этой конструкции слова **begin** и **end** выполняют роль операторных скобок. Составной оператор можно включать в любое место программы, где допускается использование только одного оператора. В свою очередь, любой из операторов составного оператора также может быть составным. Извне составного оператора нельзя передавать управление внутрь его (оператором **goto**).

4.3.2.2. Условный оператор

Условный оператор позволяет на определенном этапе выбрать одно из двух действий в результате анализа некоторых условий. Существуют следующие виды записи условного оператора:

If <условие> **then** <оператор>;
If <условие> **then** <оператор1> **else** <оператор2>;
If <условие> **then** <оператор1> **else if** <условие>
then <оператор2>
else <оператор3>;

Для условного оператора первого вида, если условие истинно, выполняется оператор, стоящий после **then**. Если же условие ложно, то этот оператор не выполняется, а выполняется оператор, следующий за условным. Например:

if x<0 **then** y=x+x.

Второй вид записи оператора позволяет производить выполнение оператора 1, если условие истинно. Если условие ложно, выполняется оператор 2. Например**:**

if x>0 **then** y:=sqrt(x) **else** y:=x.

В третьей форме записи условный оператор расширен за счет вложенности новых условий. Это приводит к сокращению числа условных операторов, но снижает наглядность программы. Новые условия могут записываться за ключевыми словами **then** и **else**. Ключевое слово **else** всегда относится к ближайшему **if**. Например:

```
if x<a then p:=ln(x)
            else if x>b then p:=sin
                        else p:=cos(x).
```

Следует помнить, что после **then** и **else** может стоять только один оператор. Поэтому, если возникает необходимость выполнения группы операторов, то их надо объединить в один, взяв в операторные скобки (т. е. использовать составной оператор **begin…end**). Кроме того, при необходимости учета нескольких условий используются логические операции: and (и), or (или), not (не).

Например, алгоритм: если A<D и A>C то Y1:=A^2 и Y2:=A*C; следует записать:

If (A<D) and (A>C) then begin Y1:= sqr(A); Y2:=A*C end; .

Операции отношения имеют наинизший приоритет по сравнению с логическими и арифметическими операциями, т. е. выполняются в последнюю очередь. При вычислении логических выражений отношения, как правило, должны заключаться в круглые скобки.

Пример 1. Точка принадлежит заданной области, если выполняется одно из неравенств **0<x<y** или **y<x<0.** Следовательно, в этом случае логическое выражение **(0<x) and (x<y) or (y<x) and (x<0)** принимает значение *true* , в противном случае – *false.*

```
Var x,y:real;
    B:boolean;
begin
  writeln ('Введите координаты x,y');
  readln (x,y);
  b:=(0<x) and (x<y) or (y<x) and (x<0);
   if b then writeln ('точка x=', x:4:2,
             ' y=', y:4:2, 'принадлежит области')
        else writeln ('точка x=', x:4:2,
             ' y=',y:4:2, 'не принадлежит области')
  end.
```

Пример 2. Вычислить скорость осаждения капелек воды в неподвижной среде электродегидратора в зависимости от значения Рейнольдса:

$$U = \begin{cases} \dfrac{d^2 g(\rho_1 - \rho_2)}{18\nu_2\rho_2}, & \text{если } 1\cdot 10^{-4} \le \mathrm{Re} \le 2.0, \\ \sqrt{3.03 dg(\rho_1 - \rho_2)/\rho_2}, & \text{если } \mathrm{Re} \le 500, \end{cases}$$

где d – диаметр капелек воды, м;
ρ_1, ρ_2 – плотности воды и нефти, кг/м^3;
ν_2 – кинематическая вязкость нефти, м^2/с.

```
Program SKOR;
 var d,g,V2,r1,r2,U,Re:real;
Begin
 writeln('Введите значение d g r1 r2 v2 Re=');
 readln(d,g,r1,r2,v2,Re);
 if (Re>=1.0E-4) and (Re<=2.0) then
 U:=sqr(d)*g*(r1-r2)/(18*V2*r2);
 if Re>500 then U:=sqrt(3.03*d*g*(r1-r2)/r2);
 writeln('U=',U);
end.
```

4.3.2.3. Оператор выбора CASE

Оператор CASE предназначен для программирования алгоритмов с большим числом разветвлений. Этот оператор обеспечивает выполнение одного оператора (простого или составного) из нескольких возможных.

Общий вид оператора CASE:

```
case <выражение–селектор> of
  <список меток 1>: оператор 1;
  <список меток 2>: оператор 2;
            . . . . .
  <список меток n>: оператор n;
else <оператор>
end;
```

Здесь значение выражения должно быть того же скалярного типа (кроме real), что и метки. Оператор выбора действует следующим образом. Если значение выражения равно одной из меток, то выполняется соответствующий ей оператор. Затем управление передается за пределы оператора выбора. Наличие символа **«;»** перед **ELSE** в конструкции этого оператора обязательно.

Замечание. Метки оператора CASE не описываются в разделе **label**, и на них нельзя переходить оператором GOTO. Метки внутри одного оператора выбора должны быть различными.

Пример. Вычислить значение теплоемкости химических веществ и соединений по выражениям:

– $C_p = a + b \cdot T$; (I)
– $C_p = a + b \cdot T + c \cdot T^2$; (II)
– $C_p = a + b \cdot T + c/T^2$; (III)
– $C_p = a + b \cdot T + c \cdot T^2 + d/T^2$. (IV)

```
Program Tepl;
 Var T,Cp,a,b,c,d:real;
     n,i:integer;
Begin
 write('введи T,a,b,c,d= ');
 readln(T,a,b,c,d);
 write('введи номер формулы n= ');
 readln(n);
    Case n of
    1:Cp:=a + b*T;
    2:Cp:=a + b*T+c*T*T;
    3:Cp:=a + b*T+c/T*T;
    4:Cp:=a + b*T+ c*T*T +d/T*T;
    end;
 writeln('Для n=',n:2,'  Cp=',Cp:7:2);
End.
```

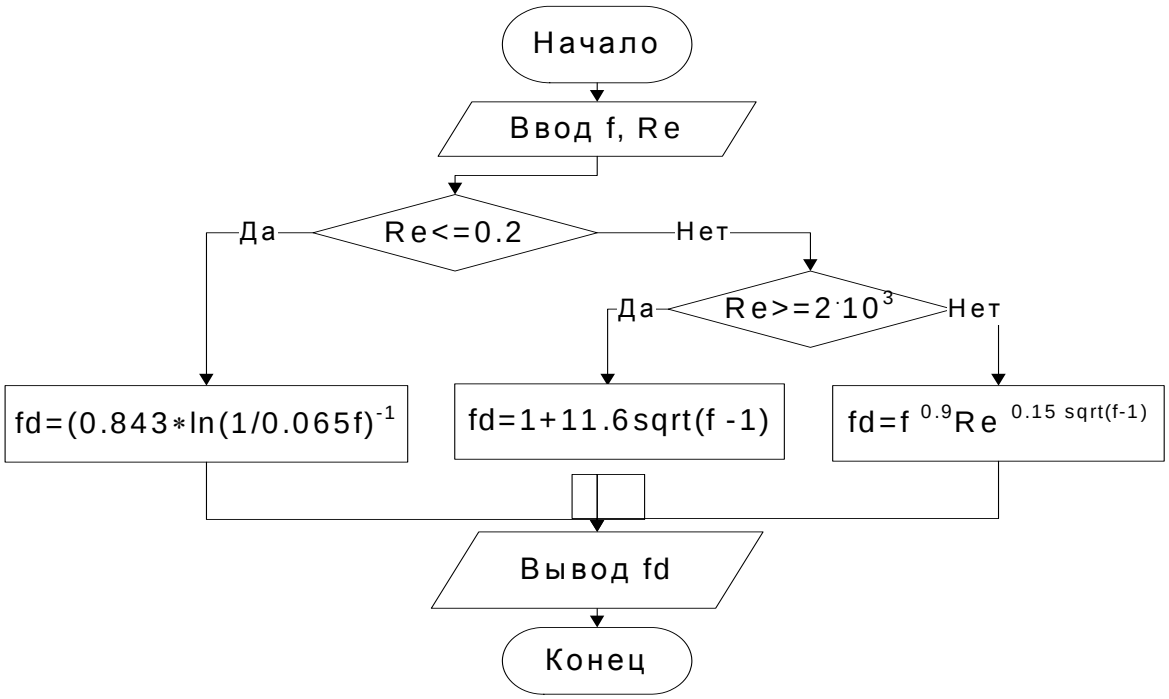
4.3.2.4. Программирование разветвляющихся алгоритмов

Идея разветвляющегося алгоритма является одной из основополагающих в программировании. Возможность создать разветвляющуюся программу обеспечивается условными операторами. Рассмотрим пример программы для разветвляющегося алгоритма.

Пример. Вычислить динамический коэффициент формы сферической частицы катализатора

$$fg = \begin{cases} \left(0.834 \ln \dfrac{1}{0.065 f}\right)^{-1}, & \mathrm{Re} \le 0.2, \\ f^{0.9}\,\mathrm{Re}^{0.15\sqrt{f-1}}, & 0.2 < \mathrm{Re} < 2 \cdot 10^3, \\ 1 + 11.6\left(\sqrt{f-1}\right), & \mathrm{Re} \ge 2 \cdot 10^3, \end{cases}$$

где f = 1.16. Обозначим $f_g = fd$.



Программа:

```
Program Koef;
 var fd,f,Re:real;
Begin
 write('Введите f,Re');
 readln(f,Re);
 if Re<=0.2 then fd:=1/(0.843*ln(1/0.065*f))
 else if Re>=2.0E3 then fd:=1+11.6*sqrt(f-1))
 else fd:=exp(0.9*ln(f))*
          exp(0.15*sqrt(f-1)*ln(Re));
 writeln('fd=',fd:10);
end.
```

4.3.2.5. Операторы цикла

Для организации циклов (повторов) при записи алгоритмов на языке Паскаль используются три вида операторов цикла:

WHILE – оператор цикла с предварительным условием;

REPEAT – оператор цикла с последующим условием;

FOR – оператор цикла с управляющим параметром.

Оператор цикла WHILE.

Общий вид оператора следующий:

while <условие> **do** <оператор>;

где <условие> – логическое выражение;

<оператор> – тело цикла (простой или составной оператор).

Оператор действует следующим образом. Проверяется условие, если оно истинно, выполняются операторы циклической части. Как только оно становится ложным, происходит выход из цикла. На литературном языке это будет выглядеть примерно так: «Пока указанное условие верно, выполнять следующее».

Пример. Зависимость удельной теплоемкости химического соединения от температуры выражается формулой

$$C_p = a + bT + cT^2,$$

где a, b, c – постоянные коэффициенты;

T – температура.

Вычислить удельную теплоемкость в интервале температур от 200 до 800 К с шагом 50 К.

Программа:

```
Program Tepl;
 var a,b,c,Cp:real;
     T,h:integer;
Begin
 writeln('Введите коэффициенты');
 readln(a,b,c);
 T:=200; h:=50;
 while T<=800 do
  begin
   Cp:=a+b*T+c*T*T;
   writeln('T=',T:3,' Cp=',Cp:7:2);
   T:=T+h;
  end;
End.
```

Действие цикла можно прокомментировать: «Пока температура меньше 800 К, вычислять значение теплоемкости». После выполнения программы все нужные нам значения теплоемкости будут выведены на экран.

Оператор цикла REPEAT.
Общий вид оператора следующий:
repeat
<оператор 1>;
. . . . . {операторы циклической части}
<оператор n>
until <условие>;

Оператор действует следующим образом. Выполняются операторы циклической части, проверяется условие. Если оно ложно, то вновь выполняется тело цикла, если оно истинно, то происходит выход из цикла. Это может быть выражено так: «Повторять действие до тех пор, пока не выполнится условие».

Примечание: так как границы цикла обозначены словами REPEAT и UNTIL, нет необходимости заключать операторы циклической части в операторные скобки **begin – end**, хотя их использование не является ошибкой.

Пример. Вычислить значение теплоемкости *Ср* с использованием оператора REPEAT.

```
Program Tepl;
 var a,b,c,Cp:real;
     T,h:integer;
Begin
 writeln('Введите a,b,c=');
 readln(a,b,c);
 T:=200; h:=50;
 repeat
  Cp:=a+b*T+c*T*T;
  writeln('T=',T:3,' Cp=',Cp:7:2);
  T:=T+h;
 until T>800;
End.
```

Примечание: действие оператора REPEAT, противоположно действию оператора WHILE, т. к. в первом условие выхода из цикла должно быть истинным, а во втором – ложным.

Значения переменных, входящих в условие операторов WHILE и REPEAT, должны обязательно изменяться в теле цикла, иначе цикл не будет завершен. (В приведенном нами примере – это значение переменной T).

Оператор цикла FOR.

Оператор цикла FOR используется для организации цикла, когда известно число повторений. Существует два варианта оператора:

– при увеличении значения параметра (цикл с положительным шагом: +1)

for i:=n1 **to** n2 **do** <оператор>;

– при уменьшении значения параметра (цикл с отрицательным шагом: –1)

for i:=n1 **downto** n2 **do** <оператор>,

где i – параметр цикла;

n1 и n2 – начальные и конечные значения параметра цикла; <оператор> – тело цикла (простой или составной операторы). Параметры i, n1, n2 должны иметь один и тот же тип, кроме real, шаг параметра цикла всегда 1.

Цикл действует таким образом. Параметру i присваивается начальное значение n1 и сравнивается со значением n2. До тех пор, пока параметр i меньше или равен конечному значению n2 (в первом варианте) или больше, или равен n2 (во втором варианте), выполняются операторы циклической части; в противном случае происходит выход из цикла.

Примечание:

1. Внутри цикла нельзя изменять начальное (n1) и конечное (n2) значения параметра цикла, а также само значение i.

2. После завершения цикла значение параметра i становится неопределенным (т. е. ничему неравным), за исключением выхода из цикла при помощи GOTO.

3. Во всех трех операторах цикла внутри цикла можно использовать операторы IF, GOTO. Разрешается в любой момент выходить из цикла, не дожидаясь его завершения. Но запрещено при помощи этих операторов передавать управление извне цикла внутрь цикла.

Пример. Рассмотрим расчет теплоемкости с использованием оператора FOR.

```
Program Tepl;
 var a,b,c,Cp:real;
     T,h,i:integer;
Begin
 write('Введи a,b,c=');
 readln(a,b,c);
 T:=200; h:=50;
 for i:=1 to 13 do
  begin
   Cp:=a+b*T+c*T*T;
   writeln('T=',T:3,' Cp=',Cp:7:2);
   T:=T+h;
  end;
End.
```

Параметр цикла i изменяется от 1 до 13, т. к. на заданном интервале температуры от 200 до 800 К с шагом 50 К должно быть вычислено тринадцать значений теплоемкости.

Следует помнить: операторы цикла WHILE и FOR могут содержать в теле цикла только один оператор. Поэтому при необходимости вычисления нескольких операторов необходимо заключать их в операторные скобки (т. е. использовать составной оператор **begin …end**).

4.3.2.6. Программирование циклических алгоритмов

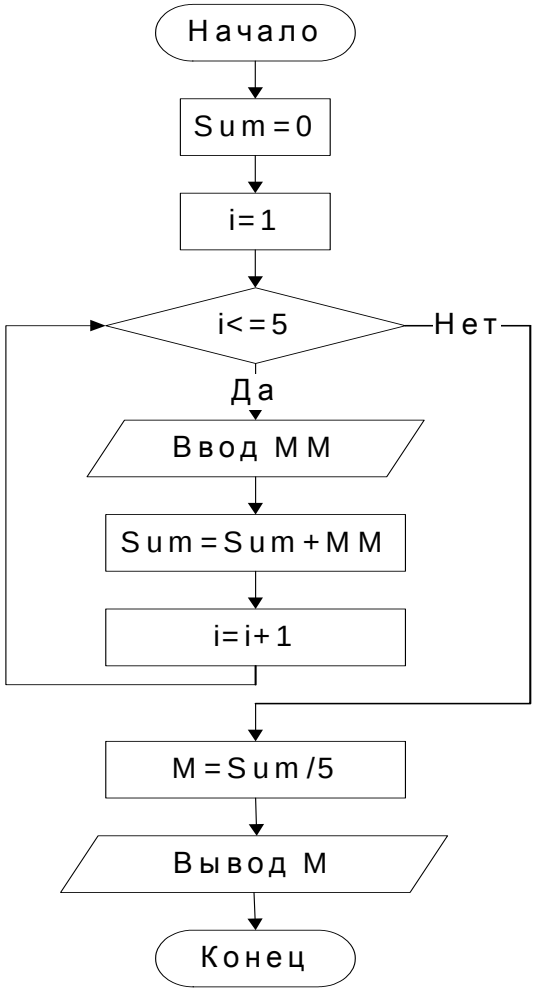
Рассмотрим примеры программ для циклических алгоритмов.

Пример 1. Составить программу вычисления средней молекулярной массы пяти органических соединений от CH_4 до C_5H_{12}.

Для этого вначале необходимо найти сумму молекулярных масс всех пяти соединений: Sum=ΣMM, а затем найти ее среднее значение M=Sum/5.

```
Program Molmas;
 var M,MM,Sum:real;
     i:integer;
Begin
 Sum:=0; i:=1;
 while i<=5 do
  begin
```

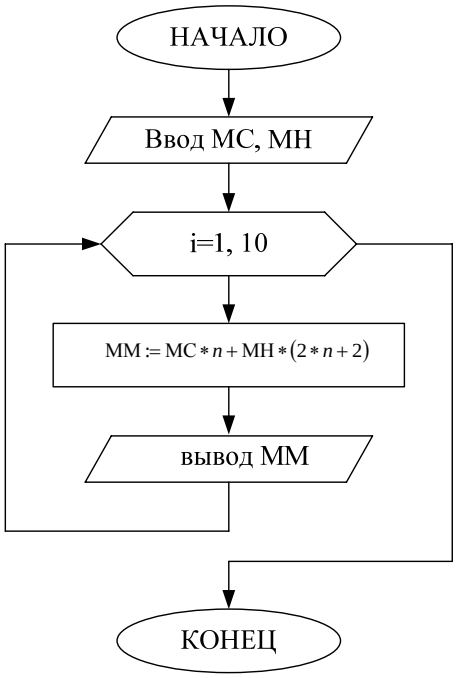
```
    writeln('Введи мол. массу');
    readln(MM);
    Sum:=Sum+MM;
    i:=i+1;
   end;
  M:=Sum/5;
  writeln('M=',M:10:3);
End.
```



Пример 2. Химическая формула углеводорода предельного гомологического ряда в общем случае имеет вид: C_nH_{2n+2}, где n – количество атомов углерода в молекуле. Вычислить молекулярную массу углеводородов от CH_4 до $C_{10}H_{22}$.

Обозначим: MM – молекулярная масса углеводорода; MC, MH – молекулярные массы углерода и водорода.

```
Program Mol;
 const MC=12; MH=1;
 var MM:real;
     n:integer;
Begin
 for n:=1 to 10 do
  begin
   MM:=MC*n+MH*(2*n+2);
   writeln('Мол. масса
       C',n,'H',2*n+2,'=',
       MM:10:3);
  end;
End.
```



В результате вычислений при каждом значении ***n*** будет выводиться на экран запись:

```
Мол. масса CH4 = 16
. . . . . .
Мол. масса C10H22 = 142
```

Пример 3. Вычислить значение константы скорости по формуле Аррениуса

$$k = k_0 \cdot \exp(-E/RT),$$

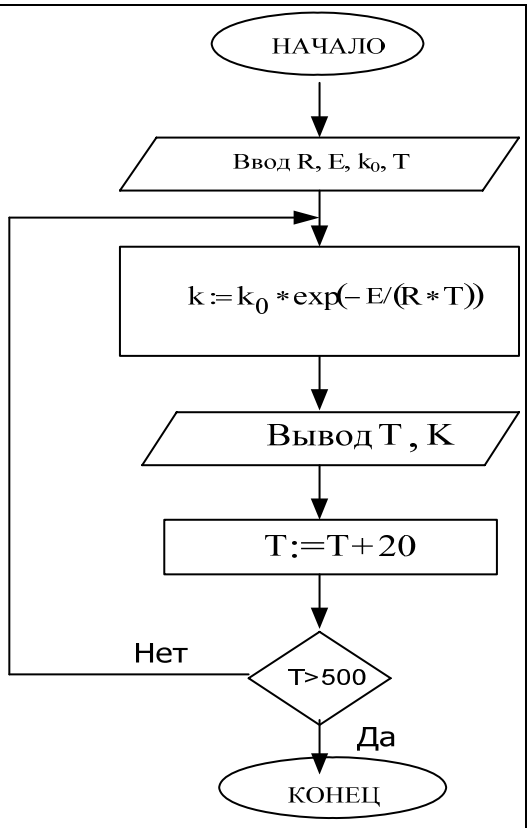
где $k_0 = 2.4 \cdot 10^6$;

$R = 0.001986$ ккал/моль · К;

$E = 12.3$ ккал/моль.

Температура изменяется от 400 до 500 К с шагом 20 К.

```
Program Constant;
 const R=0.001986;
       E=12.3;
 var k,k0:real;
     T:ineger;
Begin
 k0:=2.4E6;
readln (T);{T=400};
 repeat
k:=k0*exp(-E/(R*T));
writeln ('при T=',T:5,
         ' k=',k:10);
  T:=T+20;
 until T>500;
End.
```



4.4. Структурированные типы данных

4.4.1. Массивы

Массив – это упорядоченная последовательность элементов одного типа, обозначенных одним именем [10–17].

Отдельная величина последовательности называется элементом массива (переменная с индексом). Индекс указывает положение (адрес) элемента в массиве.

Любой массив имеет имя, размерность и длину (размер). Количество индексов у переменной с индексом определяет размерность массива. Длина массива – это общее число его элементов.

Примерами массивов могут быть:

1) вектор **x** = $\{x_1,x_2,.,x_{10}\}$ – это одномерный массив состоящий из десяти элементов x_i, где i = 1, …, 10;

2) матрица

$$a_{11}\ a_{12}\ a_{13}$$
$$a_{21}\ a_{22}\ a_{23}$$

это двумерный массив из шести элементов a_{ij}, где i = 1, 2; j = 1, 2, 3.

Описание массивов. Возможны два способа описания массивов:

1. **type** <имя типа> = **array**[<тип индексов>] **of** <тип компонент>;

var <имя массива>:<имя типа> .

Вначале определяется некоторый тип со структурой массива, а затем описывается переменная, как имеющая данный тип.

2. **var** <имя массива>**array** [<тип индексов>] **of** <тип компонент>.

Например, массивы вещественных чисел $a_1, a_2, \ldots, a_{10}$ и $b_1, b_2, \ldots, b_{10}$ можно описать:

1) **type** mas = **array**[1..10] **of** real;
   var a,b:mas;
2) **var** a,b: **array** [1..10] **of** real.

Размерность массивов в Турбо Паскале не ограничена.

Доступ к каждому элементу массива можно выполнить путем указания имени массива, за которым в квадратных скобках следует индекс элемента.

Индекс элемента может быть задан переменной – a[i]; числом – a[5]; выражением – a[2*i–1].

Примеры описания массивов:

```
var
{одномерный массив целых чисел}
x:array[1..10]of integer;
{одномерный массив вещественных чисел}
y:array[1..5]of real;
{двумерный массив вещественных чисел}
a:array[1..3,1..5]of real;
{трехмерный массив символьных данных}
b:array[1..2,1..5,1..3]of char;.
```

Ввод-вывод массивов. Для ввода-вывода массивов используются циклы. Рассмотрим ввод и вывод массивов на примерах.

Пример ввода-вывода **одномерного** массива.

Пусть для решения задачи необходимо ввести численные значения молекулярных масс десяти химических веществ: $M_1, M_2, \ldots, M_{10}$.

```
 var M:array[1..10]of real;
     i:integer;
Begin
 {ввод значений массива M в столбце}
 writeln('Введите M');
 for i:=1 to 10 do
  begin
```

```
    writeln('M',i);
    readln(M[i]);
   end;
  {вывод элементов массива M в строку}
  for i:=1 to 10 do
   write(M[i]);
End.
```

Ввод значений с экрана монитора будет происходить следующим образом. После появления записи «введи М», следует записать численное значение М[1] и нажать на «**Enter**» и так до десятого элемента включительно.

M1 <значение M[1]> ;
M2 <значение M[2]> ;
. . . . .
M10 <значение M[10]> .

Пример ввода-вывода **двумерного** массива.

Требуется ввести значения теплоемкостей пяти органических соединений, представляющих три гомологических ряда: алканы, алкены, спирты.

$Cp_{11}, Cp_{12}, \dots, Cp_{15}$
$Cp_{21}, Cp_{22}, \dots, Cp_{25}$
$Cp_{31}, Cp_{32}, \dots, Cp_{35}$

```
var Cp:array[1..3,1..5]of real;
    i,j:integer;
Begin
{ввод значений массива}
 for i:=1 to 3 do
  for j:=1 to 5 do
   begin
    writeln('Введите Cp',i,j);
    readln(Cp[i,j]);
   end;
{вывод элементов массива по строкам}
 for i:=1 to 3 do
  begin
   for j:=1 to 5 do
    write(Cp[i,j]:7:3,' ');
   writeln;
  end;
 . . . . .
End.
```

Ввод массива будет осуществляться таким же образом, как и в случае одномерного массива.

При выводе массива на экран появляются три строки по пять численных значений теплоемкостей:

```
Cp[1,1] Cp[1,2] ... Cp[1,5]
. . . . . .
Cp[3,1] Cp[3,2] ... Cp[3,5]
```

Численные значения элементов массива могут быть также заданы:

- ***в разделе*** **const**

```
Type   mas = array[1..3] of real;
       mas1 = array[1..2,1..3] of integer;
Const  M:mas = (12.3,14.6,18.4);          {одномерный массив}
       Cp:mas1 = ((10,16,8), (6,20,12));  {двумерный массив}
            или
Const  M: array[1..3] of real=(12.3,14.6,18.4);
       Cp: array[1..2,1..3] of integer = ((10,16,8), (6,20,12));
```

- ***случайными значениями от*** **0** ***до*** **1**

```
Begin    randomize;
         for i:=1 to 10 do
         M[i]:= random;
         . . . . .
End.
```

Примечание:

– **Random(x)** – возвращает случайное число от 0 до ***x***. Если функция задана без аргумента (**Random**), то будут генерироваться случайные числа от **0** до **1**;

– **Randomize –** обеспечивает (при каждом перезапуске программы) несовпадение последовательностей случайных чисел, генерируемых функцией **Random**

Программирование типовых алгоритмов вычислений приведено далее в главе 5.

4.4.2. Файлы

Удобным способом хранения информации служит запись этой информации на магнитный носитель (жесткие, гибкие диски, магнитные ленты). Запись удобна особенно тогда, когда объем информации велик и в дальнейшем предполагается использовать эту информацию в других программах. В Паскале предусмотрен специальный тип данных – фай-

лы, операции над которыми сводятся к работе с внешними носителями [8, 9, 11].

Файл – это поименованная область памяти на каком-либо носителе информации, предназначенная для хранения данных. Этим носителем может быть гибкий или жесткий диск (магнитная лента).

Внешние файлы должны быть описаны в разделе описаний программы. Описание файлов в общем случае имеет следующий вид:

type <имя файла> = **file of** <тип компонент>;
var <имя файловой переменной>:<имя типа>;

или

var <имя файловой переменной> **file of** <тип компонент>;

Файловая переменная (обозначим ее как ***f***) служит для доступа к файлу.

В Турбо Паскале существуют следующие категории файлов:

– типизированные;
– нетипизированные;
– текстовые.

В зависимости от категории объявление файлов соответственно будет:

var f1 : **file of** <тип компонент>;
f2 : **file**;
f3 : **text.**

Например:

```
var f1:file of char;
    f2:file;
    f3:text.
```

Так как по определению число элементов файла не задается, то для нахождения конца файла введена стандартная функция

Eof(var:file):boolean;

Когда наступает конец файла, Eof принимает значение *true* в противном случае – *false.*

Стандартные процедуры для работы с файлами. Работа с файлами производится посредством специальных стандартных процедур. Рассмотрим некоторые из них.

ASSIGN (f, '<имя внешнего файла>') – эта процедура связывает файловую переменную f с именем внешнего файла на диске.

Например: Assign (f1, 'fl.d'), здесь имя файловой переменной f1 связывается с файлом fl.d на диске.

RESET (f) – процедура открывает существующий файл f для чтения.

REWRITE (f) – создает и открывает новый файл для записи.

APPEND (f) – открывает существующий файл для добавления данных.

READ (f, X1,...,Xn) или READLN (f, X1,...,Xn) – считывает из файла f значения переменных X1 … Xn.

WRITE (f, X1,...,Xn) или WRITELN (f, X1,...,Xn) – записывает в файл f значения переменных X1 … Xn.

CLOSE (f) – закрывает файл f после окончания работы с ним.

Особым типом файлов являются ***текстовые файлы***. Эти файлы содержат некоторый текст, который состоит из обычных символов (например, букв алфавита и цифр). Символы текстового файла разбиты на строки.

Описание текстового файла:

Var <имя файла>: text;

Текстовый файл состоит из последовательности строк различной длины. Для определения конца строки используется функция

Eoln(var F:text) : Boolean;

Она принимает значение True, если достигнут конец строки, и значение False – в противном случае.

Для чтения из текстового файла или записи в текстовый файл можно использовать **процедуры Write (f, X1, …, Xn), Writeln (f, X1, ..., Xn), Read (f, X1, ..., Xn), Readln (f, X1, ..., Xn).**

Несмотря на то, что текстовый файл является набором символьных значений, он может использоваться (и часто используется) для хранения численных значений. При считывании или записи значений в файл происходит автоматическое преобразование из числового формата в символьный и наоборот.

Продемонстрируем работу с файлами на примере.

Пример. Составить программу пересчета концентраций химических веществ, заданных в мольных долях, в весовые:

$$BD_i = \frac{MD_i \cdot MB_i}{\sum MD_i \cdot MB_i};\ i = 1,...,5,$$

где BD_i – концентрация в весовых долях;

MD_i – концентрация в мольных долях;

MB_i – молекулярный вес веществ.

Исходные данные ввести из файла, результат вычислений поместить в файл.

Программный файл:

```
Program Conz;
 type mas=array[1..10]of real;
 var BD,MD,MB:mas;
     s:real;
     i:integer;
     f1,f2:text; {объявление файлов}
Begin
 Assign(f1,'dat'); Assign(f2,'rez');
 Reset(f1); Rewrite(f2);
 {ввод данных из файла dat}
 for i:=1 to 5 do
 read(f1,MD[i]);
 readln(f1);
 for i:=1 to 5 do
 read(f1,MB[i]);
 s:=0.0;
 for i:=1 to 5 do
 s:=s+MD[i]*MB[i];
 for i:=1 to 5 do
 BD[i]:=MD[i]*MD[i]/s;
 {вывод результатов в файл rez}
 for i:=1 to 5 do
 write(f2,BD[i]:6:2,' ');
 close(f1); close(f2);
End.
```

После написания и сохранения программного файла в новый файл, согласно последовательности ввода данных в программе (1-я строка – массив значений концентраций MD_i; 2-я строка – массив значений молекулярных весов MB_i) записывают через пробел исходные численные данные и сохраняют файл под именем **dat**.

После выполнения вычислений компьютер создает файл с именем **rez,** в который помещаются результаты расчетов (строка численных значений массива BD_i).

4.4.3. Строки

Строковые данные предназначены для обработки текстовой информации [8, 9, 11]. Данные типа string делятся на константы и переменные.

Строковые константы могут быть описаны в разделе констант и не меняются в ходе выполнения программы. Например:

```
const St='строка';
```

```
C='константа';
```

Описание строковых переменных имеет вид:

type <имя типа> = **string**[N];
var <имя переменной> : <имя типа>;

или

var <имя переменной> : **string**[N];

Здесь N – целая константа, указывающая максимальную длину строки (количество символов в строке). В Турбо Паскале 1 <N<255. Если размер строки не указан, то считается равным 255. В этом случае описание строки будет следующим:

var <имя переменной> : **string**;

Например, описание строк 'ДАВЛЕНИЕ' и 'MOL' следующее:

```
var St1:string[8];
    St2:string[3].
```

Строковые переменные аналогичны массивам типа char. Их отличие в том, что число символов (текущая длина строки) может динамически меняться в интервале от нуля до заданного верхнего значения N. Для ссылки на специальные компоненты, как обычно в случае массивов, используются переменные с индексом St1[2], St2[3]. Допускается описание строковых данных в разделе const:

```
const St1:string[8]='ДАВЛЕНИЕ';
      St2:string[3]='MOL'.
```

4.4.3.1. Операции над строками

Для строк применимы операции присваивания, сцепления и сравнения.

Операции присваивания.

```
Str1:='SR';  Str2:='SKOR';
Str1:=Str2;
```

В результате будет присвоено: Str1:='SK'. Ввод и вывод строковых переменных осуществляется без апострофов. Например, строковая переменная Str имеет значение 'TEMP'. Для выполнения оператора Readln (Str) необходимо набрать TEMP, начиная с 1-й позиции.

Замечание. Для исключения ошибок ввода-вывода строковых переменных следует всегда использовать оператор Readln (вместо Read).

Операции сцепления. Применяются для сцепления нескольких строк в одну строку. Операция сцепления обозначается знаком **+.**

Выражение	Результат
'ВК'+'12'+'15'	ВК1215

Пример

```
Program St;
 var C:char;
     S1:string[3];
     S2:string[7];
 const S1='PAS';
Begin
 readln(c); {c='L'}
 S2:=S1+'CA'+C;
 writeln('S2=',S2);
End.
```

В результате выполнения получим
S2=PASCAL.

Операции сравнения (=, <>, >, < и т. д.). Проводят сравнение двух строк и имеют более низкий приоритет, чем операции сцепления. Сравнение строк производится слева направо до первого несовпадающего символа, и та строка считается большей, в которой первый несовпадающий символ имеет больший номер в кодовой таблице ЭВМ. Результат выполнения операций сравнения над строками всегда имеет логический тип (*true* или *false*). Строки считаются равными, если они полностью совпадают по текущей (а не по объемной) длине и содержат одни и те же символы.

Например:

```
Program Pr;
 var St1:string[3];
     St2:string[8];
Begin
 St1:='Str'; St2:=St1;
 if St1=St2 then writeln('St1 равно St2')
 else writeln('St1 не равно St2');
End.
```

В результате будет напечатано:
St1 равно St2.

Следует учесть, что одинаковые заглавные и строчные буквы в строках имеют разные значения строковых переменных. Так, если St1:='STR', а St2:='Str', то St1 не равно St2.

4.4.3.2. Стандартные процедуры и функции обработки строк

Пусть переменные St, St1, St2 – переменные типа string. Рассмотрим некоторые стандартные процедуры и функции для обработки строк.

1. Delete (St, i, n) – удаляет из строки St подстроку длиной n, начиная с позиции i. Результат – новая строка без подстроки.

Например:

```
St:='TURBOBAS';
Delete (St,6,3);
```

Результат: St:='TURBO'.

2. Insert (St1, St2, i) – вставляет строку St1 в строку St2, начиная с позиции i.

Например:

```
St:='TURBO';
Insert('BAS',St,6);
```

Результат: St:='TURBOBAS'.

3. Str(val, St) – преобразует число в строковую переменную.

Например:

```
Str(1342,St1);
```

Результат: St1:='1342'.

4. Val (St, v, cod) – преобразует строковое значение в численное данное. Если во время преобразования ошибки не обнаружено, Cod=0. В противном случае Cod будет содержать номер позиции первого ошибочного символа.

5. Copy (St, i, n) – выделяет из St подстроку длиной n, начиная с позиции i.

Например:

```
St:=Copy('Turbo-Pascal',7,6);
```

Результат: St:='Pascal'.

6. Concat (St1, St2,..., Stn) – выполняет сцепление строк в указанном порядке.

Например:

```
St3:=Concat(St1,t2,St4).
```

Эта запись эквивалентна:

```
St3:=St1+St2+St4.
```

7. Length (St) – определяет длину строки.

Например:

```
St:='MOLEKULA';
L:=length(St);
```

Результат: L=8.

8. Pos (St1, St2) – определяет наименьший номер элемента в St2, начиная с которого St1 входит в St2 как подстрока.

Например:

```
St:='Давление, атм';
P:=Pos('атм',St);
```

Результат: P=10.

4.5. Подпрограммы

При решении различных задач часто возникает необходимость проводить вычисления по одним и тем же алгоритмам, но с различными данными в одной программе. Такие вычисления целесообразно оформить в виде отдельных подпрограмм. Подпрограммы являются основой модульного программирования [10–17].

В языке Паскаль имеется две разновидности подпрограмм: ***процедуры*** и ***функции***.

Структура любой подпрограммы аналогична структуре всей программы. Текст подпрограммы должен быть помещен в тексте программы непосредственно перед основным блоком (после объявления констант, типов и переменных).

Все параметры, которые используются в подпрограмме, можно разделить на две категории: глобальные и локальные.

Глобальные – это параметры, объявленные в разделе описаний основной программы, они действуют как в основной программе, так и в любой ее подпрограмме.

Локальные – это параметры, объявленные внутри подпрограммы и доступны только ей самой. Они недоступны для операторов основной программы и других подпрограмм.

Использование подпрограмм связанно с двумя этапами:

– описание подпрограмм;

– обращение к подпрограмме.

4.5.1. Процедуры

Процедуры используются в том случае, если подпрограмма имеет несколько результатов вычислений или результат является многомерной величиной, или не имеет результата.

Описание процедуры имеет вид:

Procedure <имя> (формальные параметры);
 <раздел описаний>
 label <список меток>;

const <список констант>;
type <список типов>;
var <список переменных>
Begin
<тело процедуры>
End;

и помещается в основной программе в разделе описаний.

Формальные параметры представляют собой список переменных с указанием их типа. Эти переменные не описываются в разделе описания процедур, используются только в теле процедуры и локальны по отношению к ней.

Формальные параметры подразделяются:

1. На параметры-значения – это входные данные для работы процедур и функций, в списке формальных параметров они описываются в виде

(X1:T1; X2:T2;...) или (X1,X2:T;...),

где Xi – имена параметров;
Ti – тип параметров.

Например:

```
(a,b:real; c:integer; ...)
```

2. Параметры-переменные – это, как правило, выходные данные процедуры (результаты выполнения процедуры), которые передаются в основную программу. Описание выходных параметров следующее:

```
(... var x,s:real; var y:integer)
```

В целом заголовок процедуры с описанием формальных параметров может быть таким:

```
Procedure Prim (a,b:real; c:integer;
                var x,s: real; var y:integer).
```

Процедура не может выполняться сама, ее необходимо вызвать из основной программы или другой подпрограммы по имени с указанием в скобках фактических параметров.

Обращение к процедуре:

<имя процедуры> (фактические параметры).

Например:

```
Prim (a,b,c,x,y).
```

При обращении к процедуре формальные параметры заменяются фактическими.

Следует запомнить: между формальными и фактическими параметрами должно быть соответствие по количеству, порядку следования и типу данных.

Входные фактические параметры – это те, которые передаются в процедуру. В приведенном выше примере – это параметры ***a***, ***b***, ***c***.

Входными параметрами могут быть: константы, переменные, выражения.

Выходными фактическими параметрами (которые получают значения из процедуры) могут быть только переменные (это параметры ***x***, ***y***).

Имена соответствующих формальных и фактических параметров могут быть одинаковыми или разными.

При использовании в качестве параметров процедур сложного типа данных (массивы, множества, записи) в основной программе необходимо описать имя типа этих данных, а затем указать их в списке формальных параметров процедуры.

Пример. Вычислить расход тепла на нагрев стеклобоя при варке стекла

$$Q = C_p \cdot B\,(T_м - 25) / 100,$$

где B – количество стекломассы из боя;
$T_м$ – максимальная температура варки;
C_p – средняя удельная теплоемкость стекломассы, равная

$$C_p = \frac{\left(\sum P_i A_i T_м + \sum P_i C_i\right) \cdot 4{,}19}{0{,}00146 \cdot T_м + 1},$$

где P_i – содержание окислов в стекле, i = 1, 5;
A_i, C_i – коэффициенты Шарпа и Гинтера, i = 1, 5.

Программа:

```
Program Steklo;
    type mas=array[1..5]of real;
    var P,A,C:mas;
        Cp,S1,S2,B,Q,Tм:real;
```

```
        i:integer;
Procedure Sum2(P,A,C:mas;Тм:real;varS1,S2:real);
     var i:integer;
     Begin
      S1:=0.0; S2:=0.0;
      for i:=1 to 5 do
       begin
        S1:=S1+P[i]*A[i]*Тм;
        S2:=S2+P[i]*C[i];
       end;
     end;
     Begin {основная программа}
      for i:=1 to 5 do
       readln(P[i]);
      for i:=1 to 5 do
       readln(A[i],C[i]);
      readln(Тм,B);
      Sum2(P,A,C,Тм,S1,S2); {обращение к процедуре}
      Cp:=(S1+S2)*4.19/(0.00146*Тм+1);
      Q:=Cp*B*(Тм-25)/100;
      writeln ('Cp=',Cp:10:4,' Q=',Q:10:4);
     End.
```

4.5.2. Функции

Подпрограмма функции предназначена для вычисления какого-либо одного параметра, значение которого присваивается имени функции.

Общая структура записи функции имеет вид:

Function <имя> (формальные параметры): <тип функции>;
<Раздел описания локальных переменных, меток, констант>
Begin
<тело функции>;
end;

Формальные параметры – это параметры, необходимые для выполнения данной функции, т. е. исходные данные.

Тип функции может быть только скалярным: real, integer, char, boolean, string.

Особенности функций следующие:

– функция имеет только один результат выполнения (однако может иметь несколько входных параметров);

– результат выполнения функции должен быть обозначен именем функции, т. е. внутри подпрограммы-функции должна иметь место конструкция следующего вида:

Имя функции := значение (результат вычислений).

Обращение к функции из основной программы или другой подпрограммы осуществляется непосредственно в выражении с указанием имени функции со списком фактических параметров:

<Переменная>:=<имя функции>(список фактических параметров).

Порядок обращения к подпрограмме-функции следующий.

Если при компиляции программы встречается имя подпрограммы (процедуры или функции), это имя отыскивается в описании.

В описании подпрограммы формальным параметрам присваиваются соответствующие фактические. Формальные параметры заменяются фактическими, после чего выполняется тело подпрограммы. Результат выполнения функции присваивается имени функции и передается в основную программу.

Следует запомнить: так как происходит замена формальных параметров на фактическиие, то число, тип и порядок следования формальных и фактических параметров должен обязательно совпадать.

Пример. Составить программу вычисления затрат тепла на образование силикатов при варке стекла по выражению

$$QS_j = (100 - B_j) \cdot \sum_{i=1}^{n} Q_i \cdot P_i / 100,$$

где B_j – массив значений стеклобоя, j = 1, 2, …, 5;
Q_i – удельный расход тепла, i = 1, 2, …, 6;
P_i – концентрация реагентов в шихте, i = 1, 2, …, 6.

Программа:

```
Program Teplo;
 type mas1=array[1..5]of real;
      mas2=array[1..6]of real;
 var QS,B:mas1;
     Q,P:mas2;
     i,j:integer;
```

```
Function Sum(Q,P:mas2):real;
 var i:integer;
     s:real;
Begin
 s:=0.0;
 for i:=1 to 5 do
  s:=s+Q[i]*P[i];
  Sum:=s;
end;
 {Основная программа}
Begin
 for i:=1 to 6 do
  readln (Q[i],P[i]);
 for j:=1 to 5 do
  readln (B[j]);
 for j:=1 to 6 do
  begin
   {Обращение к функции Sum(Q,P)}
   QS[j]:=(100-B[j])*Sum(Q,P)/100;
   writeln ('QS', j, '=',QS[j]:10:2);
  end;
End.
```

4.6. Модули

Наличие модулей в Турбо Паскале позволяет программировать и отлаживать программу по частям, создавать библиотеки программ и данных. Набор процедур и функций, объединенных в один блок (UNIT), может компилироваться независимо от главной программы. Благодаря этому, время компиляции для больших программ существенно сокращается. Модульный принцип построения особенно важен при разработке программ расчета сложных химико-технологических процессов (ХТП), математическое описание которых представляет собой комплекс математических описаний блоков ХТП.

4.6.1. Структура модуля

Модуль состоит из следующих частей:

- заголовка модуля;
- интерфейса модуля;
- исполнительной (реализационной) части модуля;
- секции инициализации.

Все разделы модуля, за исключением секции инициализации, являются обязательными.

Структура модуля

<Заголовок модуля>

```
UNIT  <ИМЯ МОДУЛЯ>;
<Интерфейсная часть>
INTERFACE                    {начало раздела объявлений} ;
USES <СПИСОК ИСПОЛЬЗУЕМЫХ МОДУЛЕЙ>;
LABEL
CONST                        {открытые объявления}
TYPE
VAR
PROCEDURE
FUNCTION                      {только заголовки}
<Исполнительная (реализационная) часть>
IMPLEMENTATION
USES <ИСПОЛЬЗУЕМЫЕ ПРИ РЕАЛИЗАЦИИ МОДУЛИ>;
LABEL
CONST                        {собственные объявления}
TYPE
VAR
PROCEDURE
FUNCTION                  {тела процедур и функций}
<Инициализационная часть>
BEGIN
…..
END.
```

Указанная последовательность разделов обязательна.

Заголовок модуля состоит из зарезервированного слова **unit** и имени модуля. Имя модуля должно быть единственным. Модуль должен быть помещен в файл, имя которого совпадает с именем модуля, а его расширение – **.pas.**

Пример заголовка: **UNIT mod;**

Имя модуля не может состоять более чем из восьми символов.

Интерфейсная часть начинается словом **interface.** Через интерфейс осуществляется взаимодействие основной программы с модулем (модуля с модулем).

В интерфейсе указываются константы, типы, переменные, процедуры и функции, которые могут быть использованы основной программой (модулем) при вызове этого модуля.

В разделе объявления процедур и функций указываются лишь заголовки подпрограмм. Сами подпрограммы приводятся в исполнительной части.

Исполнительная (реализационная) часть начинается словом **implementation** и заканчивается словом **end.**

Эта часть включает все программы модуля, а также локальные метки, константы, типы, переменные. Раздел **uses** необязателен. Если какой-то модуль уже указан в интерфейсе модуля, то в исполнительной части его повторять не следует.

За разделами объявления локальных меток, локальных типов, локальных переменных идут описания подпрограмм модуля (тела процедур и функций).

Инициализационная часть. Если между ключевыми словами **implementation** и **end** появляется **begin,** то полученный составной оператор **begin……end** представляет раздел инициализации модуля.

Этот раздел обычно используется для открытия файлов (например, с помощью процедуры Assign) и для формирования структур данных и переменных. Например:

```
begin
 Assign (f1, Dan.dat);
end.
```

Инициализационная часть – это основной блок модуля. Операторы, приведенные в ней, выполняются после запуска программы первыми, т. е. перед операторами основного блока главной программы, в которую включен данный модуль.

Использование модуля в основной программе. Для использования модулей в программах следует их имена указать после слова **USES**.

Например**: USES crt, mod;**

После этого в основной программе можно использовать идентификаторы, указанные в интерфейсах перечисленных модулей.

Разработанный модуль помещается в файл с именем, имеющим расширение **.pas**, например **mod.pas.**

Имя модуля в заголовке (Unit mod) должно совпадать с именем файла. Модуль транслируется отдельно, получает расширение .tpu. Например, mod.tpu.

При трансляции основной программы все используемые в ней модули (tpu-файлы) подсоединяются автоматически.

Пример. Вычислить молекулярную массу смеси по формуле

$$\text{МС} = \sum_{i=1}^{4} \text{ММ}_i \cdot \text{МД}_i,$$

а также скорость реакции по выражению

$$W = k \cdot C_1 \cdot C_{2_2},$$

где ММ_i – молекулярная масса *i*-го компонента;

МД_i – мольная доля *i*-го компонента, $k = k_0 \cdot e^{\frac{-E}{RT}}$.

Для расчета средней молекулярной массы и константы скорости сформировать модуль.

Модуль:

```
UNIT MOL;                    {заголовок модуля}
INTERFACE                    {раздел интерфейса}
 Type mas=array[1..4] of real;
 Var MD,MM:mas;
     i:integer;
```

```
 {объявление функции и процедуры}
 Function K(ko,e,r,t:real):real;
 Procedure MASS(MM,MD:mas;var MS:real);
IMPLEMENTATION          {раздел реализации}
 Function K(k0,e,r,t:real):real;
  var k1:real;
      i:integer;
 begin
  k1:=k0*exp(-e/(r*t));
  k:=k1
 end;
 Procedure MASS(MM,MD:mas;var MS:real);
  var i:integer;
 begin
  MS:=0;
  for i:=1 to 4 do
   MS:=MS+MM[i]*MD[i];
 end
end.
```

Текст модуля записывается в файл с именем **MOL.pas** и транслируется.

Основная программа:

```
Program MolMas;
 Uses Mol;              {подключение модуля}
  Var MM,MD:mas;
      C1,C2,W,k0,e,r,t,ms:real;
begin
 write('Введите k0 e r t C1 C2');
 readln(k0,e,r,t,C1,C2);
 for i:=1 to 4 do
  readln(MM[i],MD[i]);
 MASS(MM,MD,MS);        {обращение к процедуре}
 writeln('Значение молекулярной массы=',
MS:7:3);
 W:=k(k0,e,r,t)*C1* C2));{обращение к функции
                          k(k0,e,r,t)}
 writeln ('Значение скорости=',W:10:5);
```

```
end.
```

Запишем программу в файл, например с именем **Skor.pas,** и запустим её на выполнение. В каталоге библиотеки, в которой вы работаете, должны находиться файлы: **Skor.pas; Mol.pas; Mol.tpu; Skor.exe.**

4.6.2. Модули Crt, Graph

Богатство алгоритмических возможностей языка Турбо Паскаль в значительной степени достигается благодаря использованию модулей. Так, все математические функции, в том числе sqr, sqrt, exp, ln и др., описаны в модуле System, который автоматически (по умолчанию) подключается при компиляции программы.

Модуль представляет собой набор констант, типов данных, переменных, процедур и функций, что позволяет использовать его как своеобразную библиотеку описаний (как правило, используются процедуры и функции).

Турбо Паскаль располагает восемью стандартными модулями. Это System, Dos, Overlay, Graph, Crt, Printer, Turbo3, Graph3. Два последних модуля предназначены для совместимости программ, написанных в версии 3.0. Файл, содержащий модуль, имеет расширение *.tpu. Все перечисленные модули (кроме Graph, Graph3, Turbo3) объединены и хранятся в файле Turbo.tpl.

Модуль System поддерживает все стандартные процедуры и функции, обеспечивает ввод-вывод данных, обработку строк, динамическое распределение оперативной памяти и ряд других возможностей Турбо Паскаля.

Модуль Dos содержит многочисленные стандартные процедуры и функции, многие из которых по своему действию эквивалентны соответствующим командам MS DOS (GetTime, DiskSize и др.).

Модуль Overlay обеспечивает поддержку оверлеев.

Модуль Crt поддерживает ряд стандартных процедур и функций, которые обеспечивают работу с экраном дисплея в текстовом режиме, управление звуком и работу с клавиатурой.

Модуль Printer содержит драйвер печатающего устройства и позволяет организовывать вывод информации на принтер.

Модуль Graph обеспечивает работу с экраном дисплея в графическом режиме.

Для того чтобы использовать модули в программах, их имена следует указать в предложении uses, всегда находящемся после заголовка программы. Например:

```
program Pr;
uses Crt, Graph;
```

... Модуль Crt.

Всю выводимую на экран дисплея информацию подразделяют на текстовую и графическую. Соответственно выделяют текстовый и графический режимы.

Модуль CRT. Обеспечивает работу с экраном монитора в текстовом режиме.

Для инициализации текстового режима используется процедура – **TextMode(Mode:word).** Выполнение этой процедуры приводит к очистке экрана и активации указанного режима. Для задания режимов экрана используются следующие константы.

Наименование константы	Значение константы	Размер экрана	Цвет символов	Вид адаптера
Bw40	0	40×25	ч/б	цветной
Co40	1	40×25	цветной	цветной
Bw80	2	80×25	ч/б	цветной
Co80	3	80×25	цветной	цветной
Mono	4	80×25	ч/б	ч/б
Font8x8	256	80×43 (EGA)	цветной	цветной
		80×50 (VGA)	цветной	цветной

Например, TextMode(2) – черно-белый текстовый режим, 25x40.

Рассмотрим некоторые процедуры и функции модуля CRT.

TextColor(Color:byte) – устанавливает цвет выводимых на экран символов.

TextBackGround(Color:byte) – устанавливает цвет фона, т. е. цвет области, которая окружает отображаемый на экране символ.

Для этих процедур определен следующий набор констант цветов:

Числовое значение	Цвет	Используется процедурами
0	Черный	TextColor, TextBackGround
1	Синий	TextColor, TextBackGround
2	Зеленый	TextColor, TextBackGround
3	Голубой	TextColor, TextBackGround
4	Красный	TextColor, TextBackGround
5	Фиолетовый	TextColor, TextBackGround
6	Коричневый	TextColor, TextBackGround
7	Светло-серый	TextColor, TextBackGround
8	Темно-серый	TextColor
9	Светло-синий	TextColor

10	Светло-зеленый	TextColor
11	Светло-голубой	TextColor
12	Светло-красный	TextColor
13	Светло-фиолетовый	TextColor
14	Желтый	TextColor
15	Белый	TextColor
128	Мерцание	TextColor

ClrScr – очищает активное окно и устанавливает курсор в верхний левый угол.

GotoXY(x,y:byte) – перемещает курсор в позицию с координатами X, Y в рамках активного окна.

WhereX – возвращает X-координату текущей позиции курсора.

WhereY – возвращает Y-координату текущей позиции курсора.

Пример. Массив X(N) напечатать на экране в виде столбцов по M элементов в каждом. Выделить элементы, превышающие по значению величину K.

```
program P1; uses Crt;
 var x:array[1..20]of real;
     i,N,M:integer; K:real;
Begin
 writeln(' N M K');readln(N,M,K);
 for i:=1 to N do
  begin write('X[',i,']=');readln(x[i]) end;
 ClrScr;
 for i:=1 to N do
  begin
   if x[i]>K then
    begin
     TextColor(14);
     TextBackGround(4);
    end
   else
    begin
     TextColor(15);
     TextBackGround(1);
    end;
   GotoXY(10*((i-1) div M)+1,(i-1) mod M+1);
   write(i:2,' ',x[i]:6:4)
  end;
 writeln
End.
```

Для работы с клавиатурой в **Crt** предусмотрены следующие функции:

KeyPressed:boolean – возвращает значение *true*, если на клавиатуре была нажата какая-либо клавиша. В противном случае эта функция возвращает значение *false*.

ReadKey:char – считывает символ с клавиатуры. Считываемый символ на экране не отображается.

Для работы со звуком в **Crt** предусмотрены следующие функции:

Sound(F:word) – генерирует звук частотой F Гц.

NoSound – отключает динамик.

Delay(Time:word) – делает задержку (приостанавливает выполнение программы) на определенное количество миллисекунд.

Модуль Graph. Модуль Graph поддерживает графический режим работы дисплея. В этом режиме любое изображение на экране дисплея синтезируется из множества мельчайших элементов, называемых *пикселями*. Каждый пиксель представляет собой светящуюся точку таких размеров, при которых промежутки между отдельными пикселями отсутствуют. Если группа смежных пикселей светится, то они воспринимаются не как совокупность отдельных точек, а как сплошной участок. Таким образом, на экране дисплея может быть синтезировано любое графическое изображение.

В графическом режиме экран дисплея разделяется прямоугольной сеткой, каждый элемент которой имеет свои координаты. Левый верхний угол экрана имеет координаты (0,0). Значение левой координаты (X) увеличивается в горизонтальном направлении слева направо. Значение правой координаты (Y) увеличивается в вертикальном направлении сверху вниз. Количество точек по горизонтали и вертикали называется разрешающей способностью.

Координаты правой нижней границы экрана можно определить, используя функции **GetMaxX и GetMaxY**.

Реализация графического режима в ПЭВМ обеспечивается благодаря наличию специальной схемы (электронная плата), называемой графическим *адаптером*. ПЭВМ может комплектоваться следующими типами графических адаптеров: CGA, VCGA, EGA, VGA, Hercules, AT&T, PC-3270, IBM-8514. Работу графического адаптера поддерживает специальная программа, называемая *драйвером*. Загрузочный модуль драйвера хранится в специальном файле с расширением bgi. Используемый адаптер может функционировать в различных режимах.

Адаптер	Режим	Разрешающая	Число

		способность монитора	цветов
1. CGA (1)	0–3	320∗200	4
	4	640∗200	2
2. EGA (3)	0	640∗200	16
	1	640∗350	16
3. Hercules (7)	0	720∗348	2
4. VGA (9)	0	640∗200	16
	1	640∗480	16

Модуль Graph подключается при помощи **USES Graph**. С момента подключения модуля становятся доступными все находящиеся в нем модули.

Процедура инициализации:

InitGraph(var GraphDriver:integer ; {тип адаптера}
Var GraphMode:integer; {режим графики}
Var DriverPath:string {путь к драйверу}) .

Рассмотрим пример инициализации (установки) графического режима:

```
uses Graph;
 var  Gd,Gm:integer;
begin
 Gd:=Detect;
 InitGraph(Gd,Gm,'');
 if GraphResult<>grOk then Halt(1);
 ...
 CloseGraph
end.
```

Графический режим инициализируется с помощью стандартной процедуры **InitGraph**. При этом переменным Gd и Gm необходимо указать номер адаптера и номер графического режима. Если переменной Gd предварительно присвоить значение константы Detect, описанной в модуле Graph (ее значение 0), то при загрузке драйвера программа выполнит автоматическое распознавание типа адаптера. При этом, если есть выбор графических режимов, устанавливается тот из них, который обеспечивает более высокое качество изображения. Третий параметр процедуры InitGraph – путь до файла с загрузочным модулем драйвера. Если путь отсутствует, то поиск этого файла будет осуществляться в текущем каталоге. Ошибки, которые могут возникать при инициализации графического режима, анализируют с помощью функции GraphResult. Для выхода из графического режима используется стандартная процедура CloseGraph. Эта процедура восстанавливает режим, существовавший до инициализации графики.

Для создания графических изображений модуль Graph предоставляет широкий набор процедур и функций.

Вывод точки и линии:

PutPixel(X,Y:integer;Color:word) – ставит на экране точку с координатами (X,Y) цвета Color.

Line(X1,Y1,X2,Y2:integer) – выводит на экран линию, соединяющую точки с координатами (X1,Y1) и (X2,Y2).

Rectangle(X1,Y1,X2,Y2:integer) – выводит на экран изображение прямоугольника с координатами диагонали (X1,Y1) и (X2,Y2).

Circle(X,Y:integer;Radius:word) – выводит на экран изображение окружности с координатами центра (X,Y) и радиусом (Radius).

Ellipse(X,Y:integer; StAngle, EndAngle, XRadius, YRadius: word) – выводит на экран изображение эллиптической дуги с центром в точке (X,Y) от начального угла StAngle до конечного угла EndAngle с горизонтальной полуосью XRadius и вертикальной YRadius. Отсчет углов осуществляется относительно горизонтальной оси в направлении против часовой стрелки (3 часа – 0; 12 часов – 90 и т. д.). Если StAngle=0, а EndAngle=360, то будет выведено изображение полного эллипса.

SetColor(Color:integer) – устанавливает цвет линий**.**

SetLineStyle(LineStyle,Pattern,Thickness:Word) – задает тип (толщину) линии. Первый параметр изменяется от 0 до 4 и определяет тип линии.

В модуле Graph описаны следующие константы:

– SolidLn - 0 – непрерывная линия;
– DottedLn - 1 – точечная линия;
– CenterLn - 2 – штрихпунктирная линия;
– DashedLn - 3 – штриховая линия;
– UserBitLn - 4 – тип линии, указываемый пользователем.

Второй параметр задает тип линии, он необходим, если первый параметр равен четырем. Третий параметр задает толщину линии:

– NormWidth - 1 – тонкая линия;
– ThickWidth - 3 – толстая линия.

Рассмотрим пример:

```
uses Graph;
 var Gd,Gm:integer;
begin
 Gd:=Detect;
 InitGraph(Gd,Gm,'c:\bp\bgi');
 rectangle(100,50,200,100);
```

```
  line(100,50,200,100);
  line(100,100,200,50);
  ellipse(150,75,0,360,50,25);
  readln;
  CloseGraph
end.
```

Заполнение областей изображения:

Bar(X1,Y1,X2,Y2:integer) – выводит на экран закрашенный прямоугольник с координатами диагонали (X1,Y1) и (X2,Y2).

Bar3D(X1,Y1,X2,Y2:integer;Depth:word;Top:boolean) – выводит на экран изображение закрашенного прямоугольного параллелепипеда, который рисуется в изометрическом изображении с глубиной Depth. Параметр Top определяет, рисовать ли верхнюю грань параллелепипеда. Значение Top выбирается из соответствующего списка констант модуля Graph. Если рисовать, Top =TopOn, если нет – Top = TopOff.

FloodFill(X,Y:integer;Border:word) – заполняет (закрашивает) ограниченную область текущем цветом. Граница закрашиваемой области высвечивается цветом, заданным в Border.

PieSlice(X,Y:integer;StAngle,EndAngle,Radius:word) – выводит на экран изображение закрашенного сектора круга, используя в качестве центра круга точку (X,Y), начального угла StAngle, конечного угла – EndAngle и радиуса – Radius. Контур сектора высвечивается текущим цветом. Если StAngle=0, а EndAngle=360°, то PieSlice выводит на экран закрашенную окружность.

Sector(X,Y:integer;StAngle,EndAngle,XRadius,YRadius:word) – выводит на экран изображение эллиптического сектора, используя в качестве центра круга точку (X,Y), начального угла StAngle, конечного угла – EndAngle, а в качестве горизонтальной и вертикальной полуосей – XRadius и YRadius. Контур сектора высвечивается текущим цветом. Если StAngle=0, а EndAngle=360°, то на экране будет выведено изображение закрашенного эллипса.

Тип и цвет закраски можно установить с помощью процедуры SetFillStyle (Pattern,Color: Word). В модуле Graph предопределены следующие типы закрасок.

Константа	Значение	Константа	Значение
EmptyFill	0	LtBkSlashFill	6
SolidFill	1	HatchFill	7
LineFill	2	XhatchFill	8
LtSlashFill	3	InterleaveFill	9
SlashFill	4	WideDotFill	10
BkSlashFill	5	CloseDotFill	11

Вывод текстовой информации:

OutTextXY(X,Y:integer;Text:string) – выводит строку, начиная с точки, имеющей координаты (X,Y).

SetTextJustify(Horiz,Vert:word) – устанавливает значения выравнивания текста. Для установки значения выравнивания в модуле Graph определены следующие константы:

а) горизонтальное выравнивание:

- LeftText = 0 – выравнивание слева;
- CenterText = 1 – выравнивание по центру;
- RightText = 2 – выравнивание справа;

б) вертикальное выравнивание:

- Bottom Text = 0 – выравнивание снизу;
- CenterText = 1 – выравнивание по центру;
- TopText = 2 – выравнивание сверху.

SetTextStyle(Font,Direction,CharSize:word) – устанавливает текущий шрифт, тип и коэффициент увеличения символов. Параметр Font – тип шрифта. Для установки типа шрифта в модуле Graph описаны следующие константы:

- DefaultFont = 0 – побитовый шрифт;
- TriplexFont = 1 – тройной шрифт;
- SmallFont = 2 – малый шрифт;
- SansSerifFont = 3 – гротесковый шрифт;
- GothicFont = 4 – готический шрифт

и другие.

Direction задает направление вывода (0 – горизонтальное, слева направо, 1 – вертикальное, снизу вверх). CharSize – коэффициент увеличения символов.

Построение графиков. График, как и любое графическое изображение, строится в Турбо Паскале с помощью стандартных процедур модуля Graph, которые были описаны выше. Однако при этом возникают сложности, заключающиеся в следующем. Допустим, мы строим график функции $y=\sin x$ на интервале $[-\pi,\pi]$. Поскольку современные мониторы имеют разрешение 640 точек по горизонтали на 480 точек по вертикали (VGA), то очевидно, что если мы не увеличим график, он будет располагаться в верхнем левом углу и практически не будет виден. Кроме того, часть точек, имеющих отрицательную координату x или y, вообще не будут отображаться на экране. Есть и еще одна проблема: нумерация точек на экране монитора по вертикали ведется сверху вниз, в то время как при построении графика значения y должны увеличиваться снизу вверх.

Таким образом, при построении графика возникает задача масштабирования изображения.

Рассмотрим рис. 26. Здесь внешняя рамка обозначает экран монитора, внутренняя – ту область экрана, в которой должен быть нарисован график. Координаты x_1, x_2 – границы интервала, на котором строится график (в данном случае $x_1 = -\pi$, $x_2 = \pi$), y_1, y_2 – пределы изменения функции на выбранном интервале (в данном случае $y_1 = -1$, $y_2 = 1$), Nx1, Nx2 – левая и правая границы рамки, Ny1, Ny2 – верхняя и нижняя границы. Для составления программы нам потребуются формулы, которые по известным значениям x и y позволяют определить Nx и Ny – координаты точки на экране монитора. В общем виде эти формулы могут быть записаны следующим образом:

$$\mathrm{N}x = \Delta\mathrm{N}x + \mathrm{M}x \cdot (x - x_1)\ ; \qquad \mathrm{N}y = \Delta\mathrm{N}y - \mathrm{M}y \cdot (y - y_1),$$

где ΔNx и ΔNy выражают смещение графика вдоль осей OX и OY;

Mx, My – масштабные коэффициенты осей OX и OY;

знак « – » во второй формуле выражает изменение направления оси OY.

Значения ΔNx и Mx могут быть найдены из условий: при $x=x_1$ Nx = Nx_1, при $x = x_2$ Nx = Nx_2 · ΔNx = Nx_1, а Mx = (Nx_2 – Nx_1) / ($x_2 - x_1$).

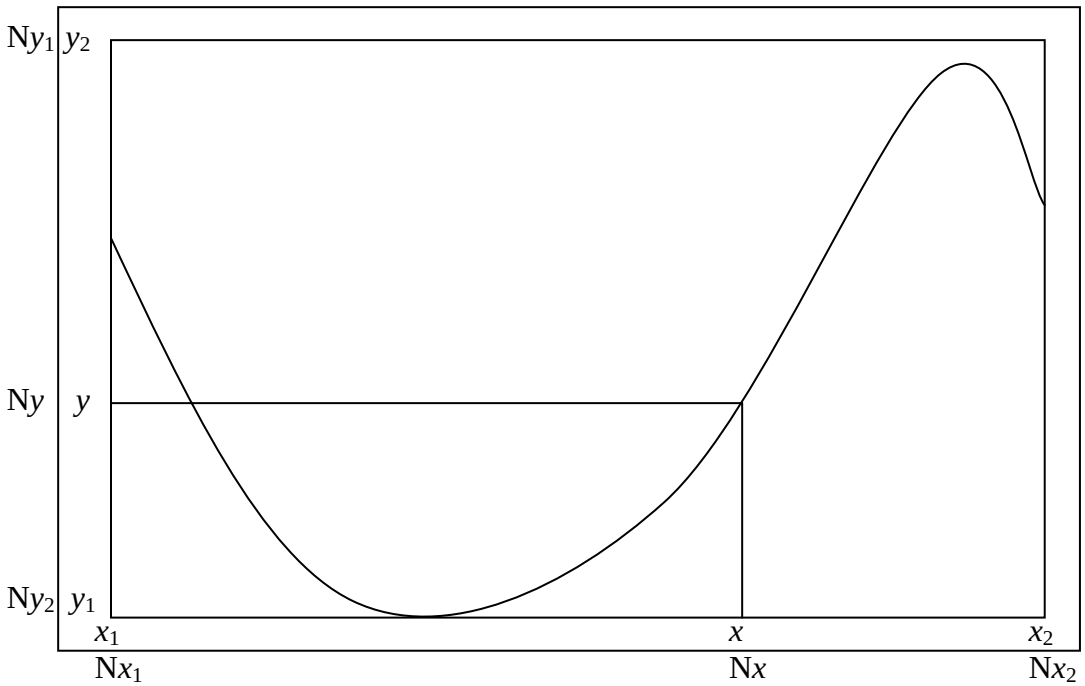


Рис. 26. График функции $y = \sin x$

Значения ΔNy и My могут быть найдены из условий: при $y=y_1$ $Ny = Nx_2$, при $y=y_2$ $Ny = Ny_2 \cdot \Delta Ny = Ny_2$, а $My = (Ny_2 - Ny_1) / (y_2 - y_1)$.

Пример программы построения графика функции $y = \sin x$ приведен ниже.

```
uses Graph;
{Описание функции y=sin(x)}
function f(x:real):real;
begin f:=sin(x) end;
var
 gd,gm,Nx,Nx1,Nx2,Ny,Ny1,Ny2:integer;
 x,x1,x2,h,y,y1,y2:real;
BEGIN

{Переход в графический режим}
 gd:=Detect;
 InitGraph(gd,gm,'c:\bp\bgi');
 x1:=0; x2:=4*Pi; h:=0.01; y1:=-1; y2:=1;
 {Размеры рамки}
 Nx1:=10; Nx2:=GetMaxX-Nx1;
 Ny1:=10; Ny2:=GetMaxY-Ny1;
 {Зарисовка рамки}
 rectangle(Nx1,Ny1,Nx2,Ny2);
 {Построение графика по точкам}
 x:=x1;
 repeat
 y:=f(x);
 {Вычисление координаты точки
  экрана по заданным  x и y}
Nx:=round(Nx1+(Nx2-Nx1)*(x-x1)/(x2-x1));
 Ny:=round(Ny2-(Ny2-Ny1)*(y-y1)/(y2-y1));
 PutPixel(Nx,Ny,12);
 x:=x+h
 until x>x2;
 readln; {Пауза}
 closegraph {Выход из графического режима}
 END.
```

4.7. Программирование типовых алгоритмов вычислений

Вычисление суммы и произведения. Пусть требуется вычислить сумму значений некоторой последовательности

$$s = a_1 + a_2 + ... + a_{20} = \sum_{i=1}^{20} a_i ,$$

где a_i – массив исходных данных.

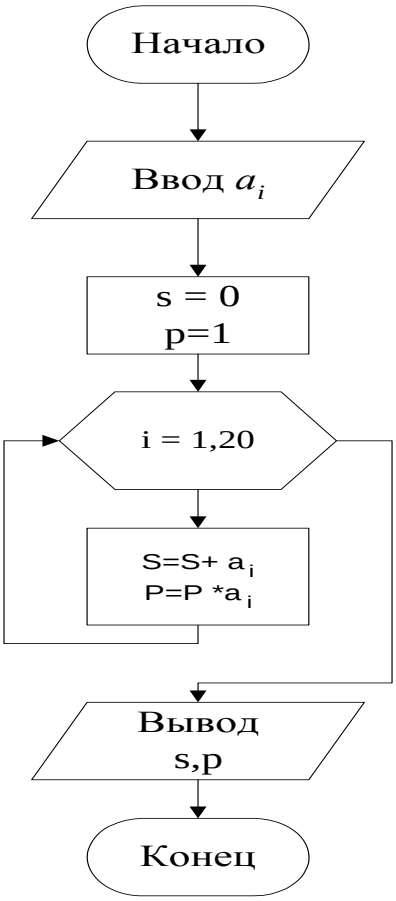
При вычислении суммы используется прием накопления по выражению

$$s = s + a_i, \quad i = 1, 2,..., n.$$

По данному выражению каждое новое значение вычисляется добавлением полученного слагаемого к сумме предыдущих. Начальное значение суммы принимается равным нулю (s = 0). При первом выполнении цикла (i=1) вычисляется значение s = 0 + a[1], на втором шаге (i = 2) – значение s = (0 + a[1]) + a[2] или s = s + a[2]. В результате повторения этой операции 20 раз получим искомую сумму.

Аналогично вычисляется произведение $z = a_1 \cdot a_2 \cdot ... \cdot a_n = \prod_{i=1}^{20} a_i$, но с той разницей, что для его накопления используется выражение $P = Pa_i$, а начальное значение произведения должно быть равно единице (P=1).

Приведем схему алгоритма и программы вычисления суммы и произведения.




```
Program SumPr;
var a:array[1..20]of
real;
    i: integer;
    P,s: real;
Begin
 for i:=1 to 20 do
  begin
   writeln('Введите
a',i);
   readln(a[i]);
  end;
 s:=0.0; p:=1;
 for i:=1 to 20 do
 begin
 s:=s+a[i];
 p:=p*a[i];
 end;
 writeln('s=',s:10,
         ' p=',p:10);
End.
```

Пример. Дана последовательность значений b_1, b_2, …, b_{10}. Вычислить сумму отрицательных элементов массива, превышающих значение переменной ***m*** и произведение положительных элементов данного массива.

```
Program Prim1;
var b:array[1..10]of real;
    i: integer;
    P,s,m: real;
Begin
   writeln('Введите m');
   readln(m);
 for i:=1 to 10 do
  begin
   writeln('Введите b',i);
   readln(b[i]);
  end;
 s:=0; p:=1;
 for i:=1 to 10 do
 begin
```

```
if (b[i]>0) and (b[i]>m) then  s:=s+b[i];
if (b[i]<0) then   p:=p*b[i];
 end;
 writeln('s=',s:8:2,' p=',p:8:2);
End.
```

Вычисление n-факториала:

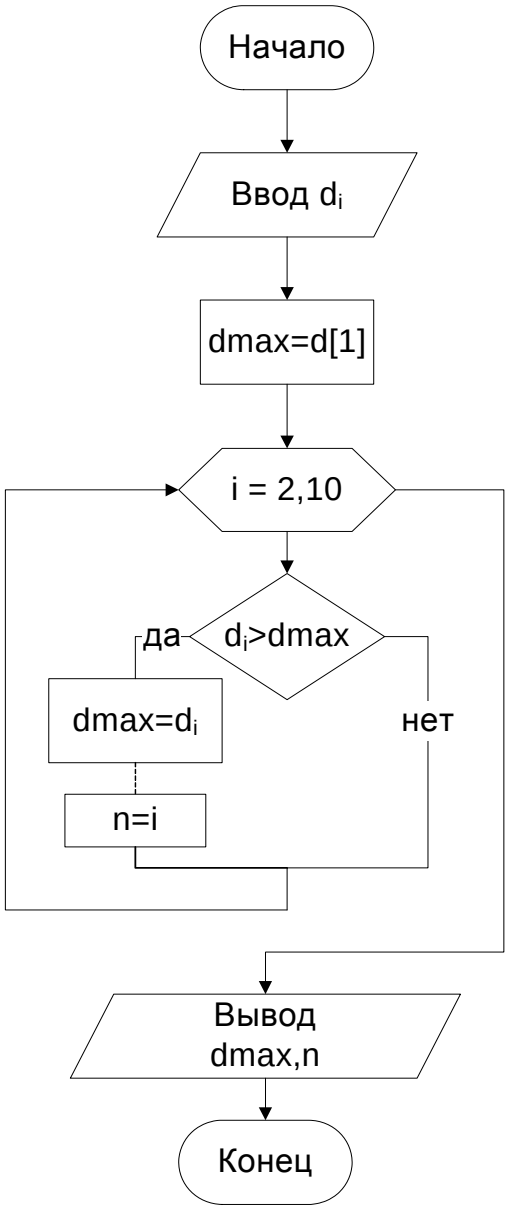
Дано целое число n=15, вычислить **n!** (т. е. $1 \cdot 2 \cdot 3 \cdot \ldots \cdot 15$).

```
     f:=1;
for i:=1 to 15 do
     f:=f*i;
```

Нахождение наибольшего (наименьшего) значения и порядкового номера рассмотрим на примере.

Пример. В массиве плотностей десяти химических элементов (d_1, d_2, ..., d_{10}) найти элемент с наибольшей плотностью (d_{max}) и его порядковый номер (n).

Нахождение наибольшего (наименьшего) значения из последовательности чисел осуществляется при помощи алгоритма попарного сравнения. В качестве начального значения наибольшего (наименьшего) принимается первый элемент массива ($d_{max}=d_1$), с которым сравнивается все остальные элементы массива. Если сравниваемый член последовательности больше d_{max}, то d_{max} присваивается его значение, в противном случае d_{max} остается без изменения. Можно за начальное приближение d_{max} принять число, которое заведомо меньше (больше) всех элементов массива, например 10^{-5} (d_{max}=1E-5).




```
Program Max;
 var d:array[1..10]of
                      real;
     dmax:real;
     i,n:integer;
Begin
 for i:=1 to 10 do
  readln(d[i]);
 dmax:=d[1];
 for i:=2 to 10 do
  if d[i]>dmax then
   begin
    dmax:=d[i];
    n:=i
   end;
 writeln('dmax=',
 dmax:10:3,' n=',n:3);
End.
```

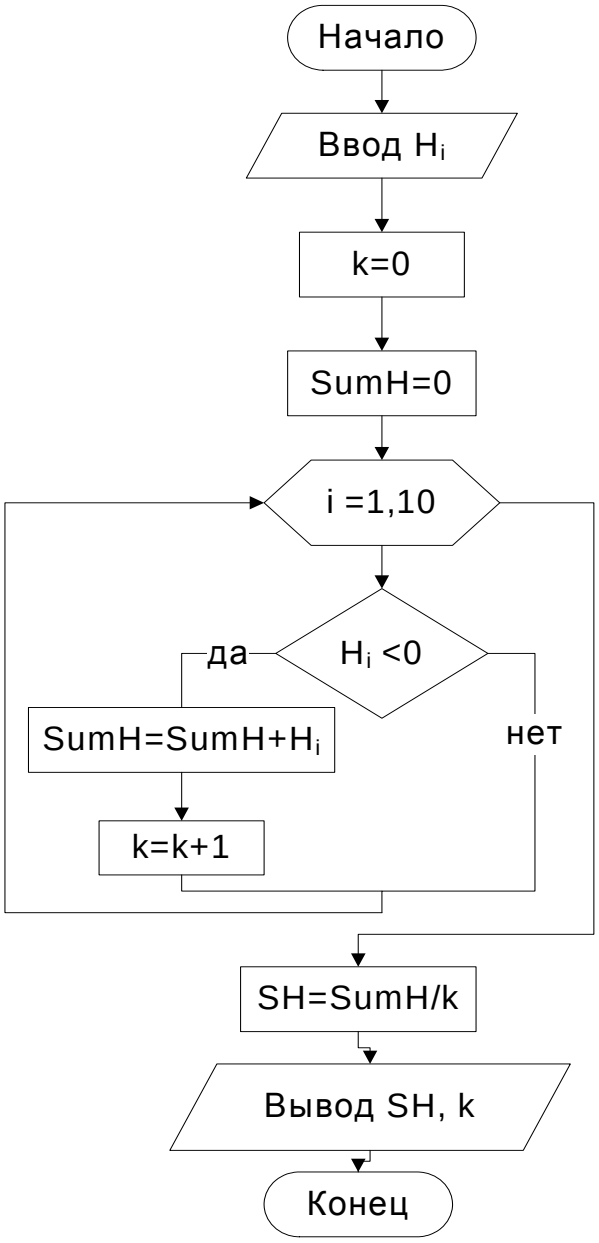
В том случае, когда ***не требуется нахождение самого значения*** максимального (минимального) элемента массива, а требуется ***определить только его номер***, алгоритм поиска номера будет следующий:

```
 Nmax:=1;
For i:=2 to 10 do
if d[i]> d[Nmax] then Nmax:= i;
```

Нахождение количества чисел. Алгоритм вычисления количества рассмотрим на примере.

Пример. Вычислить средний тепловой эффект (SH) экзотермических реакций (H1, H2, ..., H10) химико-технологического процесса.

Для того, чтобы найти средний тепловой эффект, необходимо найти суммарный тепловой эффект (SumH) и количество реакций (n) с отрицательными значениями ΔHi .




```
Program Tepl;
var H:array[1..10]of
                      real;
    SumH,SH:real;
    i,k:integer;
Begin
 for i:=1 to 10 do
  readln(H[i]);
 SumH:=0; k:=0;
 for i:=1 to 10 do
  if H[i]<0 then
   begin
    Sum:=SumH+H[i];
    k:=k+1;
   end;
 SH:=SumH/k;
 writeln('SH=',SH:10;
         ' k=',k:3);
End.
```

Количество элементов можно также вычислить, используя стандартную функцию **inc(k):**

```
for i:=1 to 10 do
  if H[i]<0 then inc(k);
writeln ('k=',k:3);
```

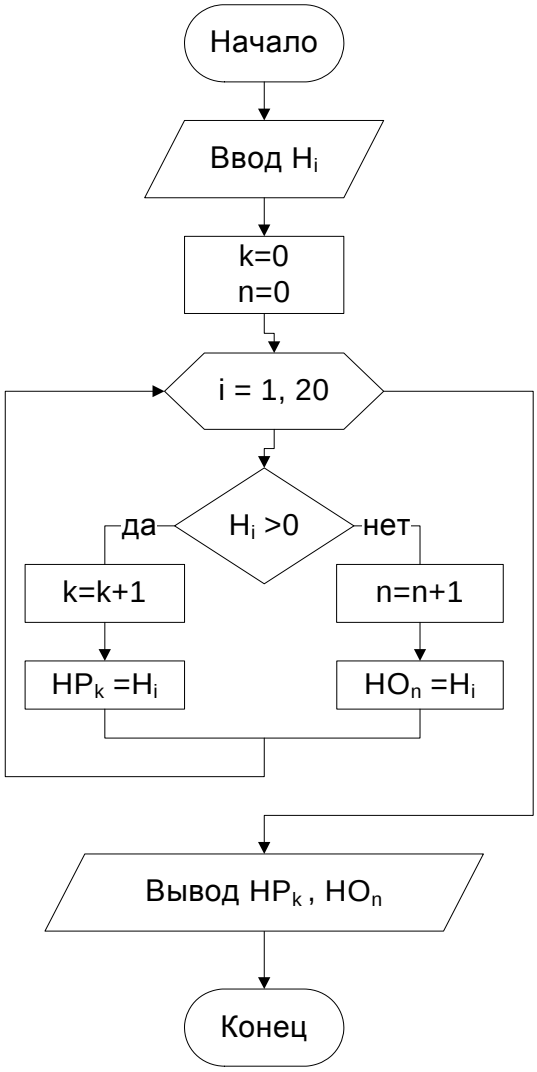
При решении практических задач часто возникает необходимость вычисления *четных* или *нечетных* элементов той или иной последовательности. В связи с этим рассмотрим пример использования функции **odd(x).**

Пример. Вычислить сумму и количество элементов массива $C_1, C_2, ..., C_{15}$, стоящих на нечетных местах, и произведение положительных элементов, расположенных на четных местах.

```
. . . . . .
S:=0; P:=1;
for i:=1 to 15 do
begin
 if odd(i) then begin inc(k); S:=S+C[i] end;
 if (C[i]>0)and not(odd(i))then P:=P*C[i];
end;
 writeln('k=',k:3,' S=',S:7:2,' P=',P:7:2);
. . . . . .
```

Преобразование массивов также рассмотрим на примере.

Пример. Дан массив энтальпий экзо-и эндотермических реакций образования химических соединений: $H_1, H_2, ..., H_{20}$. Сформировать массив ($HP_1, HP_2, ..., HP_k$), состоящий из положительных значений энтальпий, и массив, состоящий из отрицательных значений энтальпий ($HO_1, HO_2, ..., HO_n$). Обозначим: k, n – количество положительных и отрицательных элементов в массивах HO и HO соответственно.

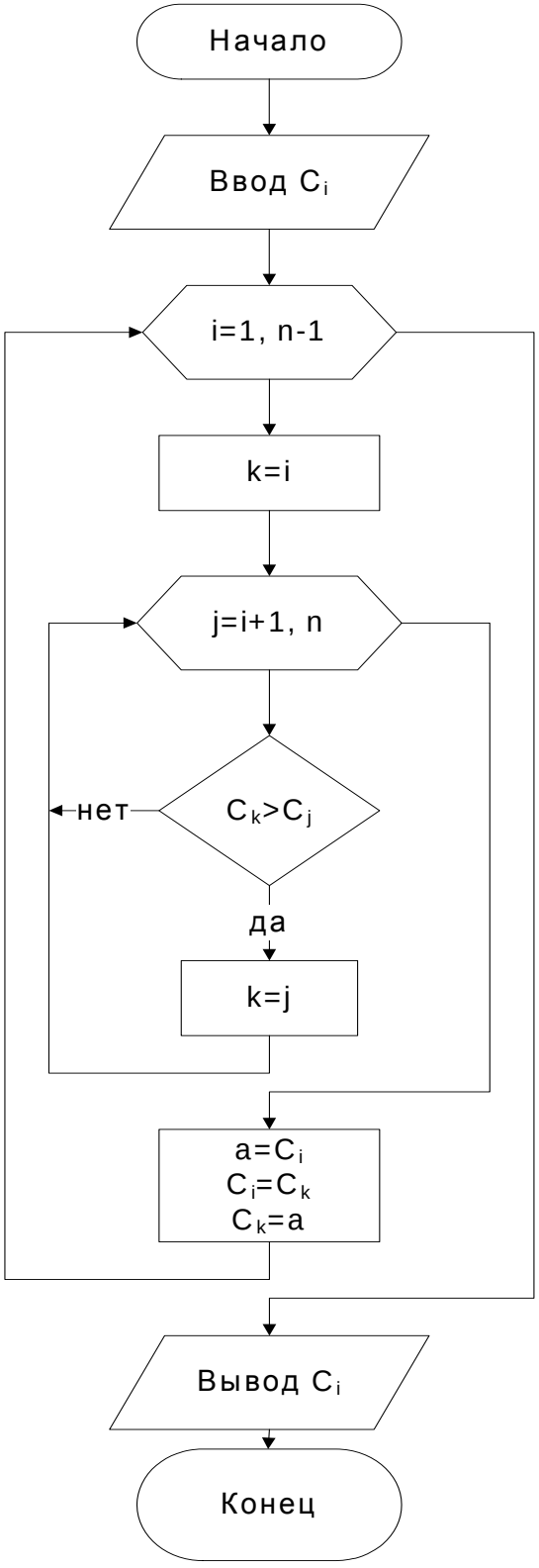



```
Program Entalp;
type mas=array[1..20]of
                        real;
var H,Hp,Ho:mas;
    k,n,i:integer;
Begin
 for i:=1 to 20 do
  readln(H[i]);
 k:=0;n:=0;
 for i:=1 to 20 do
  if H[i]>0 then
   begin k:=k+1;
    Hp[k]:=H[i];
   end
  else
   begin n:=n+1;
    Ho[n]:=H[i];
   end;
 for i:=1 to k do
  writeln(Hp[i]:10,' ');
 writeln;
 for i:=1 to n do
  writeln(Ho[i]:10,' ');
End.
```

Сортировка данных методом отыскивания наименьшего элемента. Одной из наиболее важных и часто выполняемых на ЭВМ задач является сортировка данных, т. е. расположение элементов в соответствии с заданным порядком. Известно достаточно большое количество методов сортировки. Рассмотрим один из них: сортировку методом отыскивания наименьшего элемента.

Пример. Дана смесь углеводородов C_1, ..., C_5. Расположить компоненты в порядке возрастания по числу атомов углерода в молекуле.

Массив исходных данных: 4 3 1 5 2.




```
Program Sort;
 var C:array[1..5]of
                    integer;
     a,i,j,k:integer;
 const n=5;
Begin
 for i:=1 to n do
  readln(C[i]);
 for i:=1 to n-1 do
  begin
   k:=i;
   for j:=i+1 to n do
   if C[k]>C[j] then
k:=j;
   a:=C[i];
   C[i]:=C[k];
   C[k]:=a;
  End;
 for i:=1 to 5 do
  writeln(C[i]);
End.
```

Обозначим: i – место, на которое устанавливается очередной минимальный элемент; j – текущий номер элемента в усеченной последовательности; k – номер минимального элемента.

В последовательности чисел отыскивается наименьший элемент C_k и ставится на первое место. Первый элемент ставится на место минимального (т. е. C_1 и C_k меняются местами). Затем в усеченной последовательности C_2, ..., C_n отыскивается следующий наименьший элемент и ставится на второе место и т. д. (n–1) раз. Самый максимальный сдвигается в конец последовательности.

Сортировка осуществляется по следующей схеме:

4 3 1 5 2

1 3 4 5 2

1 2 4 5 3

1 2 3 5 4

1 2 3 4 5 .

Примеры алгоритмов работы с двумерными массивами

Вычисление суммы элементов матрицы:

$$S = \sum_{i=1}^{n} \sum_{j=1}^{m} b_{ij} .$$

```
s:=0.0;
for i:=1 to n do
 for j:=1 to m do
  s:=s+b[i,j];
```

Вычисление суммы диагональных элементов матрицы (следа матрицы):

$$S = \sum_{i=1}^{n} b_{ii} .$$

```
s:=0.0;
for i:=1 to n do
 s:=s+b[i,i]
```

Вычисление суммы элементов строки матрицы:

$S_i = \sum_{j=1}^{m} b_{ij}$, i = 1,..., n (количество строк).

```
for i:=1 to n do
 begin
  s[i] :=0;
  for j:=1 to m do
   s[i]:=s[i]+b[i,j];
 end;
```

Вычисление суммы элементов столбца матрицы:

$S_j = \sum_{i=1}^{n} b_{ij}$, j = 1, ..., m (количество столбцов).

```
for j:=1 to m do
 begin
  s[j]:=0;
  for i:=1 to n do
   s[j]:=s[j]+b[i,j];
 end;
```

Нахождение наибольшего (наименьшего) значения элемента в матрице и его номера:

```
max:=b[1,1];
for i:=1 to n do
 for j:=1 to m do
  if b[i,j]>max then
                  begin max:=b[i,j];
                        n:=I; m:=j;
                  end;
```

Нахождение наибольшего элемента в строке матрицы:

```
for i:=1 to n do
 begin
  max[i]:=b[i,1];
  for j:=1 to m do
  if b[i,j]>max[i] then max[i]:=b[i,j];
 end;
```

ЛИТЕРАТУРА

1. Информатика. Базовый курс: учебник для вузов / под ред. С. В. Симоновича. – СПб., 2000. – 640 с.
2. Информатика: учебник / под ред. Н. В. Макаровой. – М.: Финансы и статистика, 1997. – 768 с.
3. Громов А. И., Сафин М. Я и др. Основы информатики и вычислительной техники: учебное пособие. – М.: изд. УДМ, 1991. – 88 с.
4. Савельев А. Я., Сазонов В. А., Лукьянов С. Э. Персональный компьютер для всех. Кн. 2. Подготовка и редактирование документов: практическое пособие для вузов. – М.: Высш. шк., 1991. – 190 с.
5. Фигурнов В. Э. IBM PC для пользователя. – Уфа: Партнерская компания «Дегтярев и сын», 1993. – 300 с.
6. Ахметов К. С. Курс молодого бойца. – М.: ТОО фирма «Компьютер Пресс», 1996. – 380 с.
7. Кенин А. М., Печенкина Н. С. Работа на IBM PC. – М.: АО «Книга и бизнес», 1992. – 368 с.
8. Миллер М. Использование Windows 98 : пер. с англ. – Киев; М.; СПб.: Вильямс, 1998. – 336 с.
9. Андердал Б. Самоучитель Windows 98. – СПб.: «Питер», 1998. – 368 с.
10. Леонтьев Б. Как установить и настроить Microsoft Windows 98. –М.: «Познавательная книга +», 1998. – 192 с.
11. Немнюгин С. А. Turbo Pascal. Учебник. – СПб., 2000. – 491 с.
12. Программирование в среде Turbo PASCAL 6.0: справочное пособие / Ю. С. Климов, А. И. Касаткин, С. М. Мороз. – М.: Высш.шк., 1992. – 160 с.
13. Офицеров Д. В., Старых В. А. Программирование в интегрированной среде Turbo Pascal 6.0. – Минск: Белорусь, 1992. – 240 с.
14. Алексеев В. Е. и др. Вычислительная техника и программирование. – М.: Высш. шк., 1991. – 400 с.
15. Епанешников В., Епанешников А. Программирование в среде Турбо Паскаль 7. – М., 1994. – 316 с.
16. Епанешников А. М. Программирование в среде Турбо Паскаль 7. – М.: «ДИАЛОГ-МИФИ», 1995. – 288с.
17. Турбо Паскаль 7.0. – Киев, 1995. – 448 с.
18. Васюкова Н. Д., Тюляева В. В. Практикум по основам программирования. Язык Паскаль. – М.: Высш. шк., 1991. – 160 с.

ОГЛАВЛЕНИЕ

Ольга Ефимовна Мойзес
Анатолий Васильевич Кравцов

ИНФОРМАТИКА

Часть 1

Научный редактор кандидат химических наук, доцент
Н. В. Ушева

Редактор А. А. Цыганкова

Подписано к печати
Формат 60×84/16. Бумага офсетная.
Плоская печать. Усл. печ. л. 7,38. Уч-изд.л. 6,68.
Тираж экз. Заказ . Цена свободная
Издательство ТПУ. 634034, Томск, пр. Ленина, 30.