Федеральное агентство по образованию
Ульяновский государственный технический университет
Кафедра «Информационные системы»

С.В. Краснов

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ TURBO PASCAL

Учебное пособие

Ульяновск 2004

УДК 681.3.06 (075)
ББК 32.973-01я7
К78

Рецензенты: кандидат технических наук Вагин Э. Д.;
кафедра сетей связи и систем коммутации
Военного университета связи (филиал, г. Ульяновск)

Утверждено редакционно-издательским советом университета
в качестве учебного пособия

Краснов, С. В.

К78 Программирование на языке высокого уровня TURBO PASCAL : учебное пособие / С. В. Краснов. – Ульяновск : УлГТУ, 2004. – 75 с.
ISBN 5-89146-500-0

Рассматриваются вопросы разработки программ в среде TURBO PASCAL. Уделено внимание основным понятиям, операторам ввода и вывода данных, составлению программ, реализующих ветвление, циклические процессы, работе с массивами, процедурами, файлами и записями. Ко всем изучаемым темам прилагаются контрольные вопросы для самопроверки и задания для самостоятельной работы по рассматриваемым темам.

Учебное пособие предназначено для студентов, обучающихся по специальности «Прикладная информатика (в экономике)» по дисциплине «Информатика и программирование», а также может использоваться в качестве учебного материала по аналогичным дисциплинам других высших учебных заведений.

Подготовлено на кафедре «Информационные системы».

УДК 681.3.06 (075)
ББК 32.973-01я7

ISBN 5-89146-500-0

Содержание

ВВЕДЕНИЕ

Язык программирования TURBO PASCAL с 1985 года применяется в общеобразовательных и высших учебных заведениях в качестве основного языка программирования и предназначен для овладения практическими навыками в программировании в соответствии с требованиями государственного образовательного стандарта и квалификационных требований.

В учебном пособии рассмотрены практические вопросы работы на ПЭВМ с программой TURBO PASCAL, которые позволяют в сжатые сроки овладеть навыками программирования. Материал пособия включает следующие разделы – оболочка программы TURBO PASCAL, работа с редактором, операторы ввода-вывода, циклы, массивы, подпрограммы, файлы, записи и сообщения об ошибках. Теоретический материал сопровождается контрольными вопросами и заданиями, позволяющими закрепить полученные знания.

Для овладения данным материалом необходимо использовать ПЭВМ, с целью составления программ на языке TURBO PASCAL, получения практических навыков и опыта работы с меню программы. В конце каждого раздела приведены практические вопросы.

Пособие предназначено для пользователей персональной ЭВМ, изучающих основы программирования по дисциплине «Информатика и программирование». Использование учебного пособия поможет освоить язык TURBO PASCAL в объеме программы и применить полученные знания для выполнения лабораторных и курсовых работ.

1. ОСНОВЫ РАБОТЫ В СРЕДЕ ПРОГРАММИРОВАНИЯ TURBO PASCAL

1.1. Язык программирования Pascal

Язык программирования Pascal (назван в честь выдающегося французского математика и философа Блеза Паскаля (1623—1662), разработан в 1968—1971 гг. Никлаусом Виртом, профессором, директором Института информатики Швейцарской высшей политехнической школы. Язык Pascal, созданный первоначально для обучения программированию как систематической дисциплине, скоро стал широко использоваться для разработки программных средств в профессиональном программировании.

Широкой популярности Pascal среди программистов способствовали следующие причины:

- Благодаря своей компактности, удачному первоначальному описанию Pascal оказался достаточно легким для изучения.
- Язык программирования Pascal отражает фундаментальные и наиболее важные концепции (идеи) алгоритмов в очевидной и легко воспринимаемой форме, что предоставляет программисту средства, помогающие проектировать программы.
- Язык Pascal позволяет четко реализовать идеи структурного программирования и структурной организации данных.
- Язык Pascal сыграл большую роль в развитии методов аналитического доказательства правильности программ и позволил реально перейти от методов отладки программ к системам автоматической проверки правильности программ.
- Применение языка Pascal значительно подняло «планку» надежности разрабатываемых программ за счет требований Pascal к описанию используемых в программе переменных, проверки согласованности программы при компиляции без ее выполнения.
- Использование в Pascal простых и гибких структур управления: ветвлений, циклов.

Для повышения качества и скорости разработки программ в середине 80-х гг. была создана система программирования Turbo Pascal. Слово Turbo в названии системы программирования — это отражение торговой марки фирмы-разработчика Borland International, Inc. (США).

1.2. Главное меню TURBO PASCAL

Первая строка содержит все команды главного меню. В последней строке экрана приведены основные доступные в каждый текущий момент функциональные клавиши с указанием их назначения. Рабочее поле (окно редактирования) предназначено для вывода на экран и редактирования

программы. **Окно редактирования** имеет по периметру рамку. На рамке окна указывается:

- сверху слева закрывающая кнопка;
- сверху в средине путь и имя файла;
- внизу слева указывается местоположение курсора в редактируемой программе (первая цифра – номер строки, вторая – номер колонки текста).

Закрытие окна осуществляется щелчком левой кнопки мыши по закрывающей кнопке. Переход между программами, расположенными в различных окнах, осуществляется левым щелчком мыши выбором команды **WINDOW** и команды **Next**. Синоним [F6].

Строка меню TURBO PASCAL

Строка меню **TURBO PASCAL** (TP) активизируется нажатием функциональной клавиши [F10] или левым щелчком мыши. Строка меню содержит имена следующих меню:

File (файл): позволяет выполнять все основные операции с файлами (создавать новые, загружать имеющиеся, сохранять созданные и отредактированные файлы, выводить на печатающее устройство содержимое этих файлов);

Edit (редактирование): позволяет выполнять все основные операции редактирования текста (копировать, вставлять, удалять фрагменты текста, а также восстанавливать первоначальный вариант редактируемого текста);

Search (поиск/замена): позволяет осуществлять поиск фрагментов текста и при необходимости производить замену найденного фрагмента новым;

Run (выполнение): позволяет запускать программу, находящуюся в рабочей зоне, а также при необходимости пошагово выполнять данную программу или её часть;

Compile (компилирование): позволяет осуществить компиляцию программы, которая находится в рабочей зоне;

Debug (отладка): содержит команды, облегчающие процесс поиска ошибок в программе (**Breakpoints –** точки остановки, окно отладки **Watch,** окно используемых программ, окно регистров, окно выходных результатов и некоторые другие);

Tools (сервис): позволяет выполнять некоторые программы, не выходя из TP;

Options (параметры): позволяет установить необходимые для работы параметры компилятора и TP;

Window (окно): позволяет выполнить все основные операции с окнами (открывать, закрывать, перемещать, изменять размер);

Help (справка): позволяет получить имеющуюся в системе справочную информацию.

Необходимое подчиненное меню активизируется (открывается) при помощи комбинации клавиш [Alt + клавиша первой буквы имени подчиненного меню], а также путем последовательной активизации клавиш [F10] и первой

буквы имени подчиненного меню. Выйти из подчиненного меню можно, нажав клавишу [ESC].

Рассмотрим некоторые *пункты меню*, обеспечивающие решение задачи в TURBO PASCAL.

Команда File содержит функции, управляющие работой с файлами.

New – удаление текущей программы из памяти и очистка экрана;

Open – загрузка файла с диска и переход в режим экранного редактирования;

Save - сохранение на диске текущего редактируемого файла и продолжение редактирования. Синоним команды – [F2];

Save as – запись текущего файла на диск под новым именем (можно задать также другие диск и каталог). После этой команды файл с новым именем становится текущим (это отражается соответствующей информацией в правом верхнем углу окна редактирования);

Save all – запись всех файлов;

Change dir – команда для изменения текущего каталога (здесь указывается имя диска или каталога, который до следующего изменения будет считаться текущим);

Print, Print setup – для работы с принтером;

DOS shell – временный выход в операционную систему, использовался ранее в ЭВМ, имеющих небольшую оперативную память (например, для выполнения команд удаления или переименования файлов). Возврат в Turbo-среду происходит после ввода команды Exit. Содержимое редактируемого файла не изменяется:

Exit – выход из Turbo-среды. Синоним [Alt+X].

Команда Edit активизирует встроенный редактор.

Команда Run объединяет функции и команды, управляющие трассировкой и выполнением программы.

Run – запуск программы на выполнение (при необходимости выполняется трансляция программы). По завершении работы программы происходит возврат в интегрированную Turbo-среду. Синоним [Ctrl+F9];

Step over – пооператорное выполнение программы. В отличие от Trace при обращении к процедуре или функции вход в них не производится, а они рассматриваются как один оператор. Синоним [F8];

Trace into – покомандное выполнение (трассировка) программы. Синоним [F7];

Go to cursor – выполнение программы (без трассировки) от текущей строки;

Program reset – выход из режима отладки: все точки прерывания и переменные в окне просмотра сохраняются, но по командам Run, Trace или Step выполнение начинается с начала программы. Синоним [Ctrl+F2];

User screen – показ результатов выполнения программы, выведенных на экран. Для возврата достаточно нажать любую клавишу. Синоним – [Alt+F5].

Команда Compile (компиляция) – перевод программы с языка Pascal, например **fist.pas**, в исполняемую программу в машинных кодах **fist.exe.** В этом пункте меню можно определить место записи откомпилированной программы или в оперативную память, или на магнитный диск. Например: для записи на магнитный диск выбрать **Destination Memory (Disk)** – **Disk**.

Все команды имеют собственные подменю, а некоторые - и несколько вложенных подменю. Для входа в главное меню следует нажать клавишу [F10], для выхода из него – [Esc] (СБРОС).

Вызов функций подменю осуществляется одним из трех вариантов:

- с клавиатуры с помощью клавиш управления указателем мыши: влево, вправо, вверх или вниз, и нажатием клавиши [Enter] для выполнения команды;
- с клавиатуры нажатием соответствующих горячих клавиш (в данном пособии они названы синонимами);
- с помощью мыши путем левого щелчка по соответствующему пункту меню.

1.3. Порядок работы с меню

1. Начальная настройка среды программирования:

Создать на диске С: каталог для файлов программ с именем по номеру группы.

Выполнить начальную настройку среды программирования.

Выбрать команду **Directories** (Каталоги) в меню **Options** (Параметры) главного окна, в поле ввода **EXE & TPU directories** окна **Directories** ввести имя каталога с указанием пути к созданному каталогу, например C:\132 или A:\132.

2. Создание новой программы.

Выбрать пункты **меню File, New**.

3. Загрузка имеющейся на диске программы.

Выбрать пункты меню **File**, **Open** (синоним [F3]), затем в появившемся окне (**Files**) выбрать папку, где находится нужный файл, затем щелкнуть по кнопке **Open**.

4. Выбор диска.

Выбрать пункты меню **File**, **Change Dir** в появившемся окне **Change Directory** щелкнуть двойным левым щелчком по надписи **Driver** и выбрать соответствующий диск, нужную папку, а затем щелкнуть левым щелчком мыши по кнопке [Ok].

5. Сохранение файла.

Выбрать пункты меню **File**, **Save**, синоним [F2], в появившемся окне задать имя файла, выбрать соответствующую папку, где сохранить файл, а затем щелкнуть левым щелчком мыши по кнопке [Ok].

6. Компилирование имеющейся в окне программы.

Выбрать команду **Compile**, затем выбрать пункт меню **Compile** и нажать клавишу [Enter]. Синоним – [Alt+F9].

7. Выполнение имеющейся в окне программы.

Выбрать команду **Run**, затем выбрать пункт меню **Run** и нажать клавишу [Enter]. Синоним – [Ctrl+F9].

1.4. Порядок ввода и редактирования программ

Набор строки заканчивается нажатием клавиши [Enter] для перехода указателя на новую строку.

Вставка символа – подвести указатель на нужное место и набрать недостающие символы.

Удаление символа. Подвести указатель на удаляемый символ:

- нажатием клавиши [Del] удаляется выбранный символ, а строка сжимается справа к указателю мыши;
- нажатием клавиши [Back Space] удаляется символ, стоящий слева от указателя, а строка сдвигается влево от указателя мыши.

Вставка строки. Маркер установить на конец строки, после которой вставить пустую (или на начало, перед которой вставить пустую), и нажать **Enter**.

Удаление строки. Маркер установить на нужную строку и нажать клавишу [CTRL + Y].

Работа с блоком (фрагментом) программы:

- установить указатель на начало выделяемого блока, пометить начало блока нажатием клавиш [Ctrl + KB];
- установить указатель на конец выделяемого блока, пометить конец блока нажатием клавиш [Ctrl + KK];
- перенос блока: установить указатель на место, куда необходимо перенести выделенный фрагмент, перенести выделенный фрагмент нажатием клавиш [Ctrl + KV];
- копирование блока: установить указатель на место, куда необходимо скопировать выделенный фрагмент, скопировать фрагмент нажатием клавиш [Ctrl + KC];
- удаление выделенного блока (фрагмента) нажатием клавиш [Ctrl + KY];
- сохранение выделенного блока на магнитном диске нажатием клавиш [Ctrl + KW], затем ввести имя сохраняемого блока;
- чтение сохраненного блока нажатием клавиш [Ctrl + KR], ввести имя вставляемого в программу файла;

Сохранение файла. Для сохранения файла нажать [F2] (при первом сохранении ввести имя файла, в котором будет сохранена программа).

1.5. Отладка и выполнение программ

1. Вызовите компилятор языка Turbo Pascal нажатием клавиши [Alt + F9] и откомпилируйте набранную программу (из меню выбрать команду **Compile**, затем пункт подменю **Compile**).

При отсутствии ошибок в программе после компиляции высвечивается сообщение **Compile Successful. Press any key**. Компилирование завершено успешно. Для продолжения нажмите любую клавишу.

При наличии ошибок высвечивается строка, где допущена синтаксическая ошибка, номер ошибки и краткое ее пояснение. После устранения ошибок компиляцию повторить.

2. Выполнение программы. Для запуска программы на выполнение выбрать команду **Run**, затем выбрать пункт меню **Run** и нажать клавишу [Enter].

Синоним – [Ctrl+F9].

3. Просмотр результатов выполнения программы [Alt+F5].

4. Возврат в редакционное окно осуществляется нажатием любой клавиши.

1.6. Завершение работы с TURBO PASCAL

Завершить работу с TP можно с помощью комбинации клавиш [Alt + X] или команды **Quit** меню **File** (кратко – [Alt + F], [Q]).

Если возникла необходимость временно выйти из TP, например, для ввода команд в ответ на подсказку MS-DOS, вызовите команду **File/DOS Shell**. При этом TP останется в памяти, но управление будет передано DOS. После выхода из TP Вы можете ввести команды DOS или запустить другие программы. Когда Вы будете готовы вновь вернуться в TP, наберите в командной строке команду **EXIT** и нажмите клавишу [Enter]. При этом TP появится в том же состоянии, в котором была, когда Вы выходили из нее.

1.7. Команды редактора текста

В табл. 1.1 перечислены клавиши и комбинации клавиш для управления курсором, вставки и удаления символа и строки, операций с блоками, поиска и замены.

Таблица 1.1

Управление курсором

Клавиши	Действие
[Home]	Курсор переводится на начало строки
[End]	Курсор переводится на конец строки
[Ctrl+Home]	Курсор переводится на первую строку экрана
[Ctrl+End]	Курсор переводится на последнюю строку экрана
[PgUp]	Продвижение по файлу на одну страницу назад
[PgDn]	Продвижение по файлу на одну страницу вперед
[Ctrl+PgUp]	Курсор переводится в начало файла
[Ctrl+PgDn]	Курсор переводится в конец файла
[Ctrl+W]	Экран сдвигается «вверх» по тексту (при этом курсор неподвижен)

Клавиши	Действие
[Ctrl+Z]	Экран сдвигается «вниз» по тексту (при этом курсор неподвижен)
[Ctrl+Q]+[B]	Курсор переводится в начало блока
[Ctrl+Q]+[K]	Курсор переводится в конец блока
[Ctrl+Q]+[P]	Курсор перемещается на исходную позицию после поиска
[Ctrl+P]	Ввод специального символа

Вставка и удаление

Клавиши	Действие
[Del]	Удаление символа, указываемого курсором
[Ins]	Переключение между режимами вставки и замены
[Backspace]	Удаление символа слева от курсора
[Ctrl+T]	Удаление слова справа от курсора
[Ctrl+Q]+[Y]	Удаление части строки от курсора до конца строки
[Ctrl+Y]	Удаление строки, указываемой курсором
[Ctrl+Q]+[L]	Восстановление строки, удаленной комбинацией клавиш [Ctrl+Y], в том месте текста, где она была расположена
[Ctrl+N]	Вставка строки

Операции с блоками

Клавиши	Действие
[Shift+стрелки]	Расширение маркируемого блока
[Ctrl+K]+[B]	Указание начала маркируемого блока
[Ctrl+K]+[K]	Указание конца маркируемого блока
[Ctrl+K]+[T]	Маркирование слова
[CtrI+K]+[H]	Снятие/восстановление маркировки
[Ctrl+K]+[I]	Сдвиг маркированного блока вправо
[Ctrl+K]+[U]	Сдвиг маркированного блока влево
[Ctrl+K]+[C]	Копирование блока в то место, где установлен курсор
[Ctrl+K]+[V]	Перенос маркированного блока в то место, где установлен курсор
[Ctrl+Ins]	Копирование маркированного блока в буфер промежуточного хранения (Edit/ Copy)
[Shift+Del]	Перенос блока в буфер промежуточного хранения (Edit/ Cut)
[Shift+Ins]	Копирование маркированного блока из буфера промежуточного хранения в то место, где установлен курсор (Edit/Paste)
[Ctrl+K]+[Y]	Удаление маркированного блока
[Ctrl+K]+[P]	Печать маркированного блока (File/Print)
[Ctrl+K]+[R]	Вставка текста из файла в позицию, указываемую

	курсором (Read)
[Ctrl+K]+[W]	Запись блока в файл (Write)
[Ctrl+O]+[F]	Переключатель режима заполнения, позволяющий оптимизировать заполнение интервалов между словами (пробелами /табуляциями)(Options/Env. /Editor/Optimal Fill)

Поиск и замена

Клавиши	Действие
[Ctrl+Q]+[F]	Поиск указанной строки (Search/Find). Для указания опций открывается специальное окно
[Ctrl+Q]+[A]	Поиск указанной строки и замена (Search/Replace). Для указания опций открывается специальное окно
[Ctrl+Q]+[[ ]	Поиск разделителя ({, [, (, ", '), парного по отношению к указываемому курсором (удобно искать границы комментариев)
[Ctrl+Q]+[]]	Поиск разделителя (},],), ", '), парного по отношении к указываемому курсором (удобно искать границы комментариев)
[Ctrl+K]+[n]	Установка отметки в тексте, 0 — цифра от 0 до 9
[Ctrl+Q]+[n]	Перевод курсора на отметку в тексте, и — цифра от 0 до 9
[Ctrl+Q]+[W]	Перевод курсора на позицию, где дано сообщение об ошибке
[Ctrl+L]	Продолжение поиска/замены с установленными ранее опциями
[Ctrl+U]	Прерывание поиска/замены

1.8. Структура программы на языке TURBO PASCAL

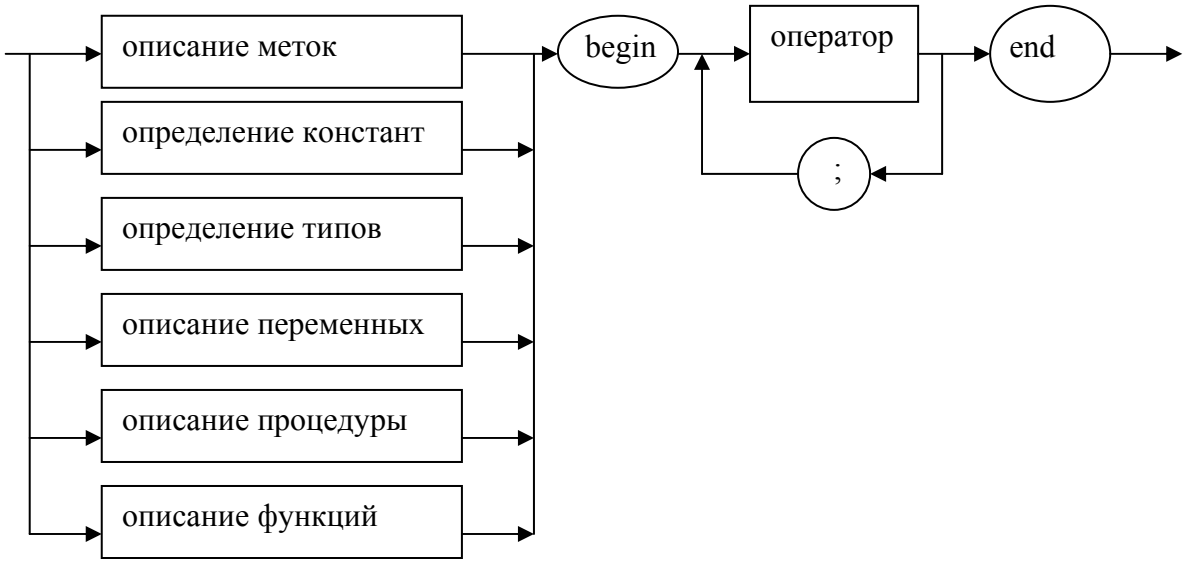
Программа реализует алгоритм решения задачи. В ней программист записывает последовательность действий, выполняемых над определенными данными с помощью определенных операций для реализации заданной цели. Основные характеристики программы: точность полученного результата, время выполнения и объем требуемой памяти.

Программа на языке **Pascal** состоит из строк. Набор текста программы осуществляется с помощью встроенного редактора текстов системы программирования **Turbo Pascal** или любого другого редактора формата DOS.

Максимальный размер программы ограничен. Компилятор позволяет обрабатывать программы и библиотечные модули, в которых объем данных и генерируемый машинный код не превышают 64 Кбайт каждый. Если программа требует большего количества памяти, следует использовать библиотечные модули (;TPU-файлы) или оверлейные структуры.

Блочная структура обеспечивает структуризацию программ на уровне исходных текстов. В идеальном случае программа на языке **Pascal** состоит из

процедур и функций, которые вызываются для выполнения из раздела операторов основной программы.



Синтаксическая диаграмма блока:

Исходя из этого можно записать структуру программы следующим образом:

```
program <имя>;
  uses <имя1, имя2,...>;
  label ...;
  const ...;
  type ...;
  var ...;
  procedure <имя>;
    <тело процедуры>
  function <имя>;
    <тело функции>
begin
<операторы>
end.
```

Заголовок программы несет чисто смысловую нагрузку и может отсутствовать, однако, рекомендуется всегда его записывать (на латинском регистре) для быстрого распознавания нужной программы среди листингов других программ. После заголовка следует программный блок, состоящий в общем случае из семи разделов:

- списка имен подключаемых библиотечных модулей (он определяется зарезервированным словом **uses);**
- описания меток;
- описания констант;
- определения типов данных;
- описания переменных;

- описания процедур и функций;
- операторов.

Любой раздел, кроме раздела операторов, может отсутствовать. Разделы описаний (кроме **uses**, который всегда расположен после заголовка программы) могут встречаться в программе любое количество раз и следовать в произвольном порядке. Главное, чтобы все описания объектов программы были сделаны до того, как они будут использованы.

РАЗДЕЛ USES

Этот раздел состоит из зарезервированного слова **uses** и списка имен подключаемых стандартных и пользовательских библиотечных модулей.

Формат:

uses <имя1>,<имя2>,... ;

Пример.

```
uses Crt;
```

РАЗДЕЛ ОПИСАНИЯ МЕТОК

Перед любым оператором языка **Pascal** можно поставить метку, что позволяет выполнить прямой переход на этот оператор с помощью оператора перехода **go to** из любого места программы.

Примечание. Нельзя выполнять переход на оператор в теле цикла, внутрь составного оператора.

Метка состоит из имени и следующего за ним двоеточия. Именем может служить идентификатор или цифра. Максимальная длина имени метки ограничена 127 символами. Перед употреблением метка должна быть описана. Раздел описания меток начинается зарезервированным словом **label** (метка), за которым следуют имена меток, разделенные запятыми. За последним именем ставится точка с запятой.

Формат:

label <имя,...>;

Пример.

```
label
Metka1, Metka2;
```

После записи метки в разделе операторов следует двоеточие, показывающее компилятору, что идентификатор используется как метка:

```
label
M1, M2;              {Описание меток}
begin
…
Ml: <оператор>       {Использование Ml в разделе операторов}
…
M2: <оператор>       {Использование M2 в разделе операторов}
end.
```

Если метка описана, но в разделе операторов не используется, то ошибки при этом не возникает, т. е. метки можно описывать и применять по мере расширения программы.

РАЗДЕЛ ОПИСАНИЯ КОНСТАНТ

В разделе описания констант производится присваивание идентификаторам констант постоянных значений. Раздел начинается зарезервированным словом **const,** за которым следует ряд выражений, присваивающих идентификаторам постоянные числовые или строковые значения. Выражения присваивания отделяются друг от друга точкой с запятой. Формат:

const <идентификатор> = <значение>;

Пример.

```
const
Maxind: word = 100;      {Типизированная константа}
Name = 'Петя';           {Строковая константа}
Code = $124;             {Константа — шестнадцатеричное значение}
```

РАЗДЕЛ ОПИСАНИЯ ТИПОВ ДАННЫХ

Тип данных может быть либо описан непосредственно в разделе описания переменных, либо определяться идентификатором типа. Стандартные типы не требуют описания в отличие от типов, образованных пользователем. Строго говоря, синтаксис языка **Pascal** не требует обязательного определения идентификатора типа и в последнем случае, так как тип можно задать перечислением в разделе описания переменных. Выбор описания типа зависит, таким образом, только от программиста и специфики программы.

Раздел описания типов данных начинается зарезервированным словом **type,** за которым следуют одно или несколько определений типов, разделенных точкой с запятой.

Формат записи:

type <имя типа> = <значения типа>;

Пример.

```
type
LatLetter = ('A'..'z');
Days = 1..31;
Matr = array[1..10] of integer;
```

Каждое описание задает множество значений и связывает с этим множеством некоторое имя типа. Например, в данном описании тип LatLetter определяет множество букв латинского алфавита. Days — множество целых чисел от 1 до 31, Matr — массив из 10 целых чисел.

РАЗДЕЛ ОПИСАНИЯ ПЕРЕМЕННЫХ

Каждая встречающаяся в программе переменная должна быть описана. Описание обязательно предшествует использованию переменной. Раздел описания переменных начинается зарезервированным словом **var** (variable —

переменная), затем через запятую перечисляются имена переменных и через двоеточие следуют их тип и точка с запятой. Формат:

var
<идентификатор, . . . > : <тип>;

В рассматриваемом примере программы три переменных A, B и Proizved могут принимать целочисленные значения, описаны следующим образом:

var
A,B, Proizved : integer;

РАЗДЕЛ ОПИСАНИЯ ПРОЦЕДУР И ФУНКЦИЙ

В общем случае подпрограмма имеет ту же структуру, что и программа. Для описания подпрограмм используются зарезервированные слова **procedure** и **function**, которые записываются в начале подпрограммы. Формат процедуры:

procedure <имя процедуры> {<параметры>} ;
<разделы описаний>
<раздел операторов>
end;

Формат функции:

function <имя функции> {<параметры>} : <тип результата>;
<разделы описаний>
<раздел операторов>
end;

РАЗДЕЛ ОПЕРАТОРОВ

В программе на языке **Pascal** раздел операторов является основным, так как именно в нем с предварительно описанными переменными, константами, значениями функций выполняются действия, позволяющие получить результат, ради которого создавалась программа.

Раздел операторов начинается зарезервированным словом **begin** (начало), далее следуют операторы языка, отделенные друг от друга точкой с запятой. Завершает раздел зарезервированное слово **end** (конец) с точкой.

Например:

begin {Начало программы}
Write ('Введите значение целого числа A >');{Вывод запроса на экран}
Readln (A); {Ввод значения A с клавиатуры}
Write ('Введите значение целого числа B >');
Readln (B);
Proizved := A ***** B; {Вычисление переменной Proizved}
Write ('Произведение чисел ',A,' и ',B,' = ',Proizved); {Вывод ответа}
end. {Конец программы}

Операторы выполняются строго последовательно в том порядке, в котором они записаны в тексте программы в соответствии с синтаксисом и правилами пунктуации.

Слова **begin** и **end** являются аналогом открывающей и закрывающей скобки в обычных арифметических выражениях.

КОММЕНТАРИИ

Для лучшего понимания программы в ней записывается пояснительный текст — **комментарий.** Комментарий можно записать в любом месте программы, где разрешен пробел. Текст комментария ограничен символами { } или (* *) и может содержать любые комбинации латинских и русских букв, цифр и других символов алфавита языка **Pascal**. Ограничений на длину комментария нет, он может занимать несколько строк.

Примеры.

{Начало программы} или **(*Начало программы*)**

{Вывод запроса на экран}

{Ввод значения A с клавиатуры}

В ограничителях (* *) пробелы между скобкой и звездочкой запрещены. В тексте не должны находиться знаки ограничителей, с которых комментарий начинается. Например, текст комментария {Пример {1} задания {4}} вызовет ошибку при компиляции. Однако ограничители { } можно вложить в (* *) и наоборот: (*Пример{1} задания {4} *) или {Пример (* 1 *) задания (* 4 *)}.

Комментарий игнорируется компилятором и поэтому никакого влияния на программу не оказывает. По месту положения в программе комментарии можно подразделить на четыре класса: объясняющие назначение программы, поясняющие смысл идентификаторов переменных и констант, описывающие логически обособленные части программы, объясняющие трудно понимаемые элементы алгоритма. В удачно прокомментированной программе легко найти ошибку, проанализировав различие между замыслом автора (в комментариях) и реализацией (в тексте программы).

Ограничители { } и (* *) удобно использовать при отладке программ. В процессе отладки часто требуется временно исключить выполнение какой-либо части программы. Конечно, этого можно добиться, уничтожив временно ненужные операторы или обойдя их с помощью оператора **go to**. Однако оба этих способа неприемлемы по ряду совершенно понятных причин: повторный набор вновь понадобившихся операторов, путаница с операторами **go to** и т. д. Гораздо удобнее просто заключить временно ненужную часть программы в {} или (* *), которая будет восприниматься компилятором как комментарий.

Например:

```
begin                {Начало программы}
Write ('Введите значение целого числа A >'); {Вывод запроса на экран}
Readln (A);          {Ввод значения A с клавиатуры}
C:= A * B;     {Вычисление переменной C}
{Временно невыполняемая часть программы
end.                 {Конец программы}
```

При необходимости { } или (* *) можно убрать, и программа будет выполняться в полном объеме.

БИБЛИОТЕЧНЫЕ МОДУЛИ ПОЛЬЗОВАТЕЛЯ

Понятие библиотечного модуля является одним из основных в идеологии программных систем на языке Turbo Pascal. Именно они служат средством создания библиотек подпрограмм (процедур и функций). **Библиотечный модуль** — это результат компиляции в режиме **Compile** с установленной директивой Destination = Disk одной или нескольких процедур и функций. Модуль имеет имя, при упоминании которого в разделе **uses** любой программы можно получить доступ к каждой из находящихся в нем процедур или функций.

Создание библиотечного модуля требует определенной организации с применением зарезервированных слов **unit**, **interface**, **implementation**, **begin**, **end**. Система сама определяет структуру компилируемого файла и создает соответственно .TPU-файл (при обнаружении **unit** и т. д.) или .EXE-файл (при отсутствии **unit**, **implementation** и т. д.). В первом случае формируется библиотечный модуль, во втором — готовый к выполнению по вызову из DOS загрузочный модуль.

1.9. Рекомендации по стилю программирования

Накопленный опыт программирования привел к формированию следующих рекомендаций по составлению наглядных и легко читаемых программ.

1. Стандартизация стиля программирования заключается в том, что необходимо всегда придерживаться одного способа программирования, записи программы.

2. С целью рационального размещения текста не следует операторы программы писать сплошным текстом.

Для четкого выявления вложенности управляющих структур требуется особым образом располагать операторы в тексте, так что служебные слова, которыми начинается и заканчивается тот или иной оператор, записываются на одной вертикали, а все вложенные в него операторы записываются с некоторым отступом вправо. При записи конструкций языка более глубоких уровней вложенности следует сдвигать их от начала строки вправо. Каждое описание и каждый оператор следует писать с новой строки. Продолжение описаний и операторов на новые строки надо сдвигать вправо. Следует избегать длинных строк.

3. Рекомендуется любую программу сопровождать комментариями, поясняющими назначение всей программы и отдельных ее блоков, процедур, функций.

4. Имена для объектов программы надо выбирать так, чтобы они наилучшим образом соответствовали этим объектам, отражали их назначение.

5. Списки идентификаторов в блоках описания следует упорядочивать — это облегчает поиск в них нужных элементов.

6. Программирование сверху вниз. В процессе разработки алгоритма и программы следует начинать с самой общей модели решения, постепенно

уточняя ее до уровня отдельного блока и затем детально прорабатывая каждый блок.

1.10. Пример простейшей программы

Пример простейшей программы и выполняемые действия представлены в таблицах 1.2.1. и 1.2.2.

Таблица 1.2.1

Схематичное представление прогаммы

№ пп.	Составные части ПГМ	Текст программы № 1	Устройства ЭВМ
1 2 3 4 5 6	Описательная часть программы	**PROGRAM** KRUG; **CONST** PI=3.14; VAR R:INTEGER; S:REAL;	ОП ***PI*** 3.14 ***R*** 5 ***S*** 15.7
7 8 9 10 11 12 13 14 15	Операторная часть программы	**BEGIN** WRITE (‘введи R ’); READLN (R); S := PI * R * R; **WRITELN (‘радиус круга’);** **WRITELN (‘ R = ’,R:4);** **WRITELN (‘площадь круга’);** **WRITELN (‘S = ‘, S : 5 : 1);** **END.**	МОНИТОР введи R 5 радиус круга R = 5 Площадь круга S=15.7

Таблица 1.2.2

Действия выполняемые при решении задачи на ЭВМ

№ пп.	*Описание операторов*
1	PROGRAM – служебное слово начала программы KRUG – имя программы (набирать на латинском регистре)
2	CONST – начало раздела описания констант
3	PI=3.14; – запись в ОП значение константы PI
4	VAR – начало раздела описания переменных

Окончание таблицы 1.2.2

5	R:INTEGER; – Выделение в ОП места для размещения переменной R – целого типа
6	S:REAL; – Выделение в ОП места для размещения переменной S – вещественного типа
7	BEGIN – начало операторной части программы
8	WRITE (‘введи R ’); – вывод на экран комментария – введи R
9	READLN (R); – ввод значения переменной R с клавиатуры в ОП
10	S := PI * R * R; – вычисление значения S и занесения его в ОП
11	WRITELN (‘радиус круга’); – вывод на экран комментария
12	WRITELN (‘ R = ’,R:5:2); – вывод на экран R = и значение R
13	WRITELN (‘площадь круга’); – вывод на экран комментария
14	WRITELN (‘S = ‘, S : 5 : 1); – вывод на экран S = и значение S
15	END. – конец программы

Контрольные вопросы и задания

1. Что включают в себя имена данных?
2. Сколько в следующем списке зарезервированных слов:

X, Program, Y, Summa, MyMoney, Произведение, Vova, begin, end, if, repeat, Read?

3. Из каких разделов состоит программа?
4. Какие действия производятся при выполнении раздела VAR?
5. В каких случаях надо использовать переменные:
   1) если в программе используется какое-то число?
   2) если в вычислениях какой-то операнд постоянно меняет свое значение?
   3) если операнд в выражении хотя бы один раз меняет значение?
6. Какие заголовки программ правильны:
   1) program Zarplata?
   2) program Сумма?
   3) program Summa Nalogov?
   4) программа Teach_Kurs?
   5) program 12Kurs2?
   6) program Summa_Elementov?
7. Какая структура программы правильная?

1) program MyProgram;
```
begin
    Writeln ('Привет');
end.
```
2) program MyFirst;
```
begin
    X:=Y+100;
end.
```

7. Какой из перечисленных разделов обязателен в программе:
   1) раздел var?
   2) раздел const?
   3) раздел type?
   4) раздел begin .. end.?
   5) раздел label?
8. Какие из комментариев неправильны:
   1) { Программа вычисляет логарифм введенного числа};
   2) (* Это тоже комментарий *);
   3) {{ Комментарий в комментарии }};
   4) (* { И это комментарий в комментарии } *).

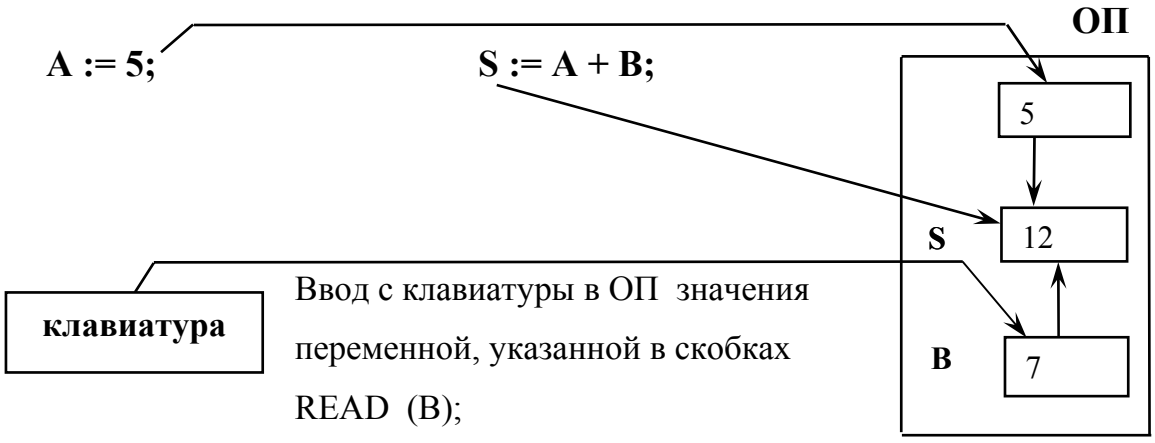
2. РАЗРАБОТКА ПРОГРАММ НА ЯЗЫКЕ TURBO PASCAL

2.1. Операторы ввода-вывода

2.1.1. Оператор присваивания

‘**S**’ **–** имя переменной, ‘**:=**’ – знак присваивания, ‘**A+B**’ **–** выражение. После выполнения такой строки в памяти ЭВМ будет записано значение или вычисленное выражение, указанное после знака присваивания.

2.1.2. Операторы ввода



После набора на клавиатуре цифры 7 и нажатия клавиши [Enter], значение переменной указанной в скобках будет занесено в ОП (более одного значения вводится через пробел или после каждого операнда нажимают клавишу [Enter]).

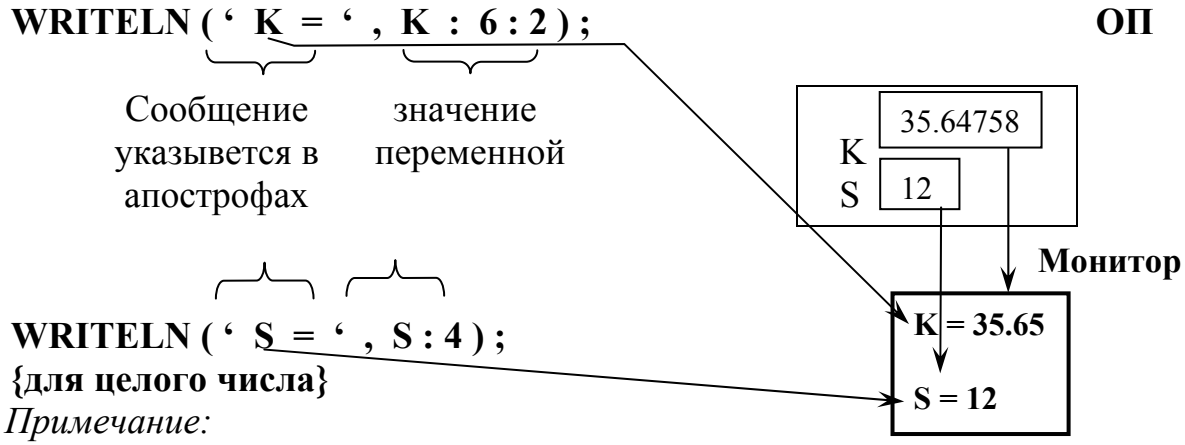
2.1.3. Оператор вывода

Пусть в ОП будут записаны значения переменных в ячейках:

S : = 12; – целое число

K: = 35.64758; – вещественное число

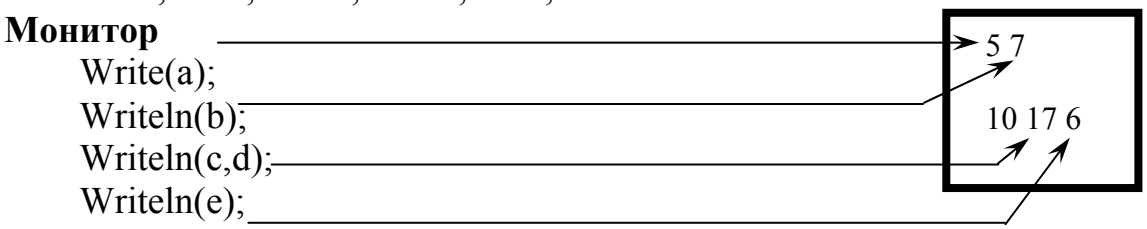
Вывод из ОП на экран сообщения и (или) значение переменной.



Примечание:

После выполнения операторов **Read** или **Write** указатель остается на месте вывода (ввода) данных, а после выполнения операторов **Readln** или **Writeln** указатель перемещается на новую строку. Пример: что будет выведено на экран после выполнения фрагмента программы ?

A:=5; b:=7; c:=10; d:=17; e:=6;



Форматированный вывод

Для целого числа после имени переменной через двоеточие указывается количество позиций, отводимых для вывода числа, например: WRITE (‘S = ‘,S : 4).

Для вещественного числа после имени переменной через двоеточие указывается количество позиций, отводимых для вывода числа, вторая цифра указывает на количество позиций после запятой, например: WRITE (‘ K = ‘ , K : 6 : 2) ;

Контрольные вопросы и задания

Вопросы:

1. Какие процедуры служат в Pascal для выполнения операций ввода-вывода?
2. Напишите оператор ввода переменной К с клавиатуры?
3. Для каких целей служит оператор присваивания.
4. Чем отличаются операторы ввода Read и Readln?
5. Для каких целей служит оператор Write?
6. Чем отличаются операторы вывода Write и Writeln?
7. Для чего в процедурах вывода определяется ширина поля вывода?
8. Какие обозначения используются в форматах вывода?

Задания:

1. Составить программу для вычисления высот треугольника со сторонами a, b, c по формулам:

$h_a=\frac{2}{a}\sqrt{p(p-a)(p-b)(p-c)}$; $h_b=\frac{2}{b}\sqrt{p(p-a)(p-b)(p-c)}$; $h_c=\frac{2}{c}\sqrt{p(p-a)(p-b)(p-c)}$,

где p=(a+b+c)/2.

2. Составьте программу вычисления площади прямоугольника по введенным в диалоге двум сторонам. Запишите текст программы на диск под именем okr.pas, откомпилируйте и проверьте ее действие.

3. Составьте программу вычисления длин высот треугольника, у которого длины сторон A, B, C.

4. Составьте программу вычисления величины силы тока на участке электрической цепи сопротивлением R Ом при напряжении U В.

5. Составьте программу вычисления напряжения на каждом из последовательно соединенных участков электрической цепи сопротивлением R1, R2, R3 Ом, если сила тока при напряжении U В составляет I А.

6. Напишите программу, которая вводит значения трех переменных A, B, C типа Real и выводит их сумму. Ввод каждого значения произвести с отдельной строки. Результат также помещается на отдельную строку. При составлении программы обеспечьте приглашение к вводу данных.

7. Составьте программу, которая выводит на экран компьютера заставку, аналогичную следующей:

```
**************************
*
*          Программа
*
*     вычисления суммы чисел
*
*        Автор: В. И. Петров
*
**************************
```

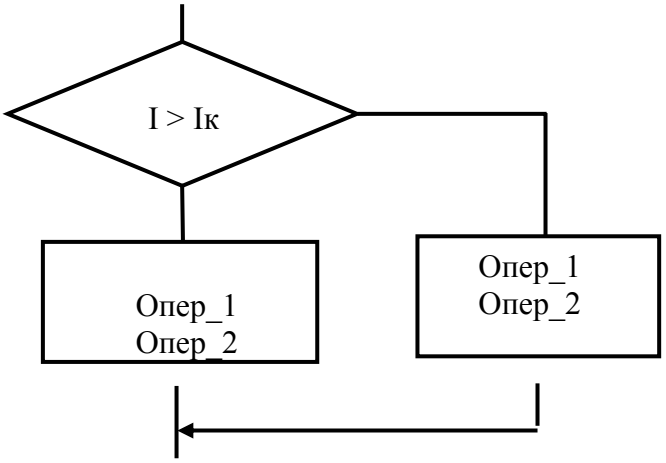
8. Напишите программу, которая вводит значения четырех переменных A, B, C, D типа integer и выводит их сумму. Ввод пары значений A и B произвести на одной строке, C и D — на другой. Результат вывести на отдельную строку и курсор оставить на той же строке.

2.2. Программирование разветвляющихся процессов

Условный оператор IF используется для изменения естественного порядка выполнения операторов программы. Если условие истина, то выполняется первая ветвь, иначе – вторая. Таким образом, условный оператор – это средство ветвления вычислительного процесса.

Составной оператор IF имеет 2 формы: полное ветвление и сокращенное.

2.2.1. Ветвление полное:




```
IF  I < = Iк  THEN
      BEGIN
        Опер_1;
        Опер_2;
      END
  ELSE
      BEGIN
        Опер_1;
        Опер_2;
      END;
```

Примечание:

1. В операторе IF перед ELSE точка с запятой не ставится.

2. Условный оператор управляет только одним оператором, поэтому, если после ключевых слов Then и ELSE требуется произвести более одного действия, то необходимо использовать операторные скобки Begin End.

3. Внутри операторных скобок после каждого оператора точка с запятой ставится.

Пример выполнения задачи на полное ветвление

Задача № 1. Вычислить корни квадратного уравнения общего вида $ax^2 + bx + c = 0$ в области действительных чисел.

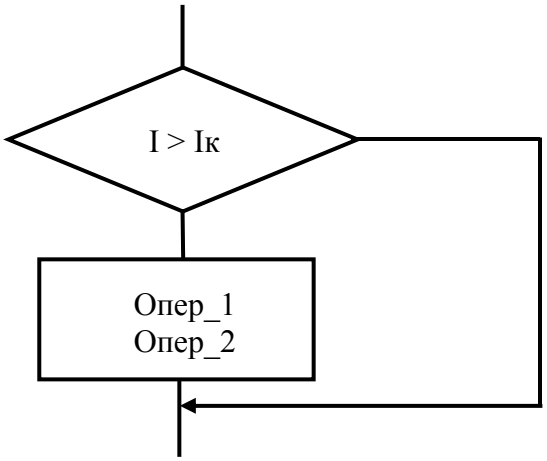
Программа имеет вид :

```
(* ОПРЕДЕЛЕНИЕ КОРНЕЙ КВАДРАТНОГО УРАВНЕНИЯ *)
PROGRAM KU;                                  {Имя программы}
 VAR                                         {Раздел описания переменных}
      A,B,C:INTEGER;                         {Коэффициенты уравнения}
      D,X1,X2:REAL;                          {Дискриминант и корни уравнения}
BEGIN
 WRITE('ВВЕДИТЕ КОЭФФИЦ. A,B,C  '); {Вывод сообщения}
 READ (A,B,C);                               {Ввод данных с клавиатуры}
 WRITELN ('A=',A,'B=',B,'C=',C);             {Эхо-печать ввода исходных
данных}
 D:=SQR(B)-4*A*C;                            {Вычисление дискриминанта}
 IF D>0 THEN                                 {Проверка выполнения условия}
     BEGIN
        X1:=(-B+SQRT(D))/(2*A);              {Выполняемые действия }
        X2:=(-B-SQRT(D))/(2*A);              {Если условие ИСТИНА}
        WRITELN ('X1=',X1,'X2=',X2);         {Вывод результата }
     END
```

```
            ELSE IF D=0 THEN            {ИНАЧЕ, Проверка выполнения
условия}
                BEGIN
                    X1:=(-B+SQRT(D))/(2*A);      {Выполняемые действия }
                    X2:=X1;                      {Если условие ИСТИНА}
                    WRITELN ('X1=',X1,'X2=',X2);
                END
                    ELSE  WRITELN ('НЕТ РЕШЕНИЯ');  {Если условие ложь}
  END.                                              {Конец программы}
```

Если вторая ветвь отсутствует, тогда имеет место сокращенное ветвление. Фрагмент программы представлен ниже.

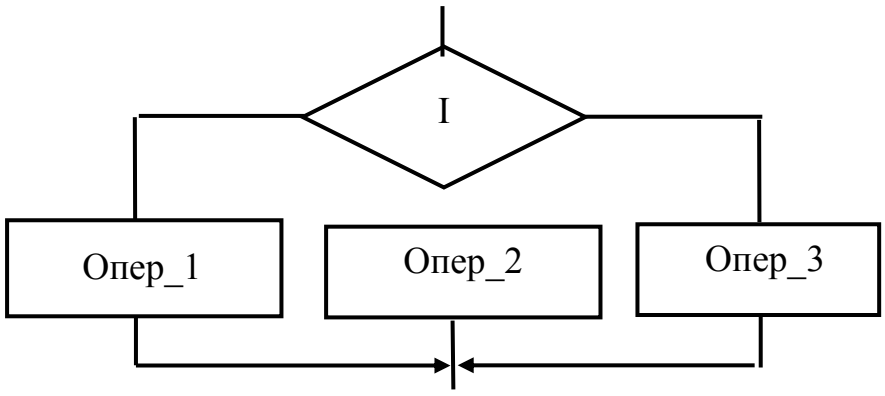
2.2.2. Ветвление сокращенное:




```
IF I < = Iк  THEN
   BEGIN
       Опер_1;
       Опер_2;
    END;
```

2.2.3. Оператор выбора:

Оператор **case** работает следующим образом. Сначала вычисляется значение выражения-селектора, затем обеспечивается реализация того оператора, константа выбора которого равна текущему значению селектора. Если ни одна из констант не равна текущему значению селектора, выполняется оператор, стоящий за словом **ELSE,** при его отсутствии выполняется оператор, стоящий за словом **end**.




```
CASE   I   OF
        1:  Опер_1;
        2:  Опер_2;
        3:  Опер_3;
     ELSE Опер 4;
END;
```

Пример выполнения задачи на использование оператора выбора

Задача № 2. Составить программу для ввода на экран монитора номера дня недели и вывода соответствующего ему дня недели на русском языке.

Программа решения задачи имеет вид:

```
PROGRAM DNED;                                  {заголовок программы}
    VAR                                        {раздел описания переменных}
      N:INTEGER,
    BEGIN
       WRITELN ('ВЫВЕДИТЕ НОМЕР ДНЯ НЕДЕЛИ');{Вывод сообщения}
       READ(N);                                {Ввод значения n с клавиатуры}
       CASE N OF                                  {Выбор варианта}
         1:WRITELN('понедельник');             { Выполняемые операторы}
         2:WRITELN('вторник');                 {в зависимости от значения
селектора}
         3:WRITELN('среда');
         4:WRITELN('четверг');
         5:WRITELN('пятница');
         6:WRITELN('суббота');
         7:WRITELN('воскресенье');
       END;                                       { Конец оператора Case}
  END.                                            {Конец программы}
```

Контрольные вопросы и задания

Вопросы:

1. Что представляет собой составной оператор? Как ограничиваются операторы, объединенные в составной оператор?
2. Назначение, формы записи и порядок выполнения оператора условия if.
3. Особенности использования вложенных условных операторов.
4. Каковы отличия оператора выбора case от оператора условия if?
5. Для чего служит ключ выбора и какого он может быть типа?
6. Сколько меток может быть перед оператором в списке выбора?

Задания:

1. Составьте программу, реализующую эпизод применения компьютера в книжном магазине. Компьютер запрашивает стоимость книг, сумму денег, внесенную покупателем; если сдачи не требуется, печатает на экране «спасибо»; если денег внесено больше, то печатает «возьмите сдачу» и указывает сумму сдачи; если денег недостаточно, то печатает об этом сообщение, указывающее размер недостающей суммы.
2. В ЭВМ поступают результаты соревнований по плаванию для трех спортсменов. Составьте программу, которая выбирает лучший результат и выводит его на экран с сообщением, что это результат победителя заплыва.
3. Ввести два числа. Меньшее заменить полусуммой, а большее – удвоенным произведением.

4. Вычислить $y = \begin{cases} \sin X, & \text{при } X>0 \\ \operatorname{tg} X, & \text{при } X<=0 \end{cases}$

5. Составить программу для вычисления значений функции:

$$y = \begin{cases} x-1 & \text{при } x \le 2; \\ x^2 & \text{при } 2 < x < 3; \\ 6 & \text{при } 3 \le x \le 5. \end{cases}$$

2.3. Программирование циклов

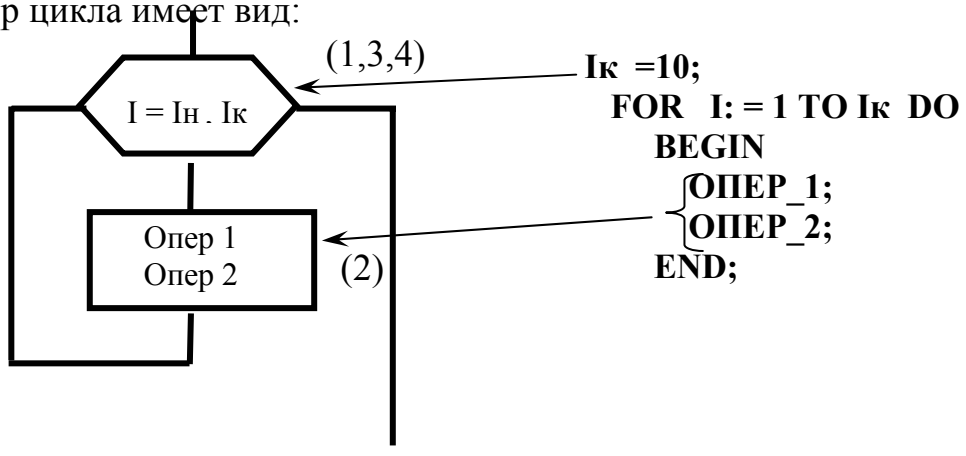
Для всех операторов цикла характерно следующая особенность. Повторяющиеся вычисления записываются всего лишь один раз. Вход в цикл возможен только через его начало. Переменные оператора цикла должны быть определены до входа в циклическую часть. Необходимо предусмотреть выход из цикла: или по естественному его окончанию, или по оператору перехода.

Цикл содержит:

1) подготовку (начало) цикла. (Управляющую переменную, ее начальное, конечное значение и шаг приращения);
2) тело цикла (повторяющиеся операторы);
3) изменение значения управляющей переменной на величину шага;
4) проверку на окончание цикла.

2.3.1. Цикл с параметром

Оператор цикла с параметром используется в тех случаях, когда заранее известно, сколько раз должна повторяться циклическая часть программы. Оператор цикла имеет вид:



При программировании циклов с параметром необходимо помнить следующие ***правила организации цикла***:

1) параметр цикла, начальное и конечное значения, должны быть одинакового типа, их тип может быть любым скалярным типом (стандартным, перечисляемым, ограниченным), кроме вещественного;

2) очередное значение параметра вычисляется автоматически, для целого типа шаг изменения значения параметра цикла равен 1 при TO и – 1 при DOWNTO;

3) запрещено изменять внутри тела цикла значение управляющей переменной цикла;

4) запрещено входить в цикл с помощью оператора GOTO, минуя оператор FOR,;

5) цикл не выполняется вообще, если начальное значение больше (при DOWNTO – меньше), чем конечное;

6) после служебного слова DO может стоять только один оператор; если в цикле нужно выполнить группу операторов, то их заключают в скобки BEGIN-END;

7) из составного оператора, входящего в оператор цикла, можно выйти до окончания этого цикла с помощью оператора GOTO, тогда последнее значение параметра цикла сохраняется.

Пример использования цикла с параметром

Задача № 3. Вычислите степень y = a действительного числа a с натуральным показателем n. Воспользуемся для вычислений следующей формулой:

До начала цикла (подготовка цикла) y:= 1, на каждом шаге цикла (для i = 1,2,...,n) – y:= y*a. Цикл с параметром i будет выполнен n раз.

Программа решения задачи имеет вид:

```
(*ОПРЕДЕЛЕНИЕ СТЕПЕНИ ВЕЩЕСТВЕННОГО ЧИСЛА*)
PROGRAM STEPEN;                {Имя программы}
  VAR                            {Раздел описания переменных}
    A,Y:REAL;
    I,N:INTEGER;
BEGIN (*STEPEN*)
  WRITELN ('ВВЕДИТЕ ЧИСЛО И СТЕПЕНЬ ЧИСЛА'); {Вывод  оообщения}
  READ (A,N);                                  {Ввод значения переменных}
  Y:=1;                                        {Ввод начального значения}
    FOR I:=1 TO N DO                         {Цикл для вычисления степени}
      Y:=Y*A;                                  {Вывод сообщения}
   WRITELN (N,'СТЕПЕНЬ ЧИСЛА',A);           {Вывод сообщения}
  WRITELN ('РАВНА',Y)                        {Вывод результата}
END.(*STEPEN*).                            {Конец программы}
```

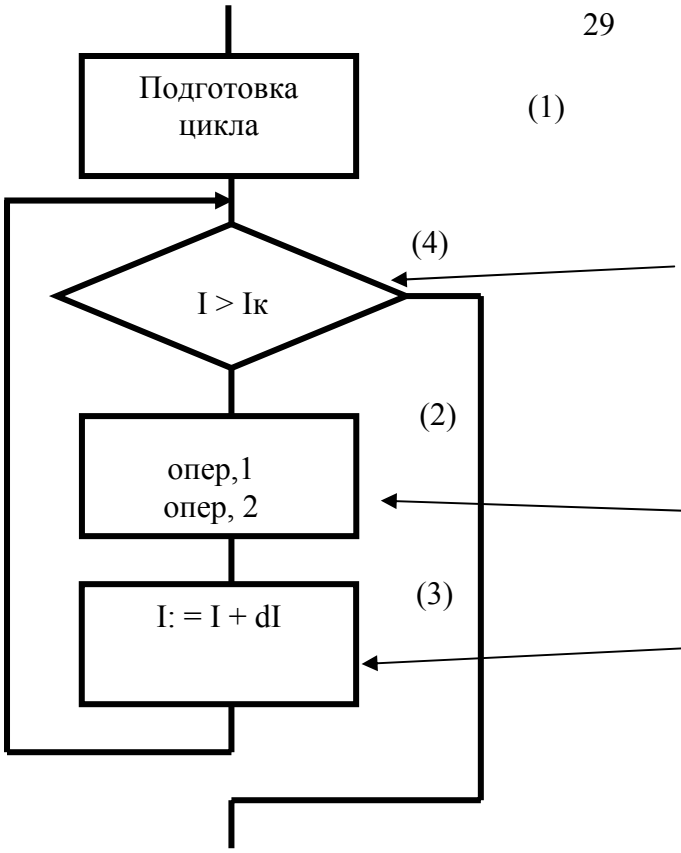
2.3.2. Цикл с предусловием

Цикл с предусловием используется, как правило, в тех случаях, когда заранее неизвестно число повторений цикла.

Форма записи оператора цикла с предусловием:

В цикле с предусловием тело цикла заключается в операторные скобки

В теле цикла с предусловием и постусловием необходимо указывать изменение управляющей переменной на величину шага.



Здесь WHILE (Пока), DO (выполнить) – служебные слова.

До начала циклов с предусловием и постусловием необходимо указывать начальное значение управляющей переменной.

Пример использования цикла с предусловием

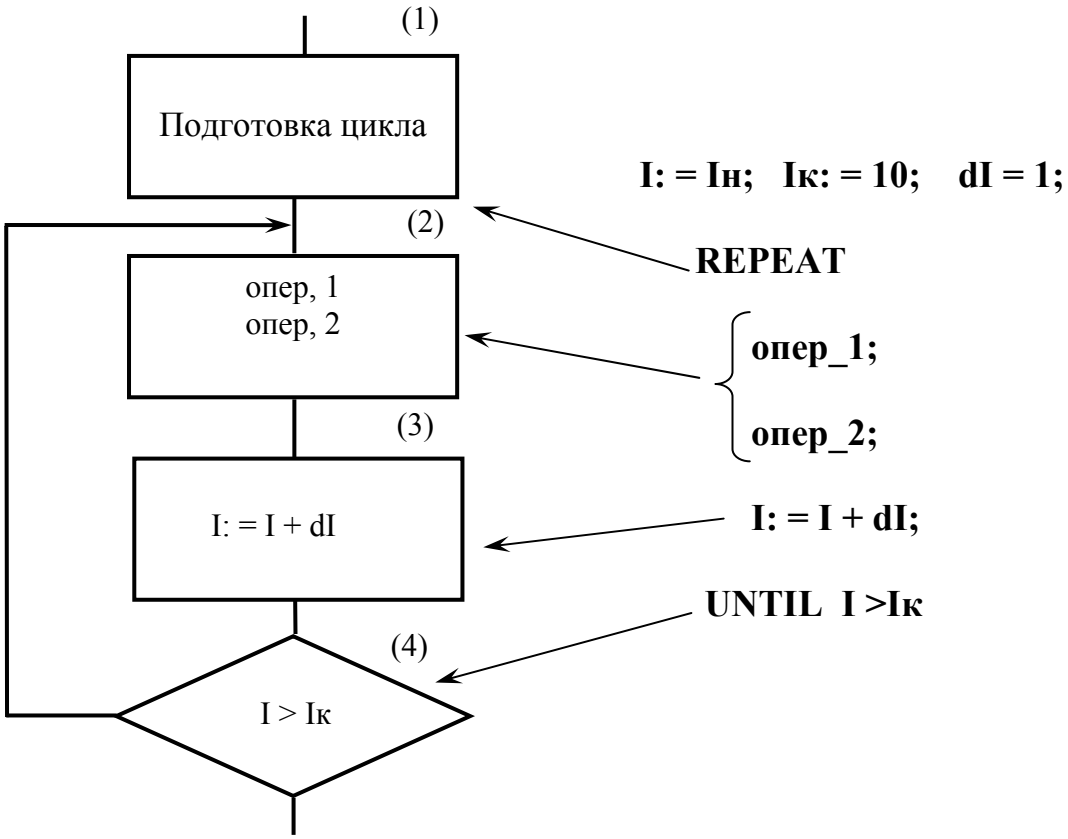
Задача № 4. Составить программу для вычисления значения функции $y = ax^2$. Переменная x изменяется от 5 до 25 с шагом 1, полученный результат вывести на экран.

Программа решения задачи имеет вид:

```
program pred5;                          {Заголовок программы}
 var                                    {Раздел описания переменных}
   x:integer;
   a,y:real;
 begin
    writeln ('Введи параметр a ');      {Вывод комментария}
     readln (a);                        {Ввод с клавиатуры значения a}
     y:=0;                              {Обнуление переменной для накапливания
суммы}
     x:=5;                              {Задание начальных условий}
       while x<=25 do                   {Начало  цикла с предусловием}
         begin;
            y:=a*x*x;                   {Тело цикла с предусловием}
            x:=x+1;                     {Изменение переменной цикла на величину
шага}
         end;                           {Конец цикла}
    writeln ('Функция равна ',y);       {Вывод результата}
 end.                                   {Конец программы}
```

2.3.3. Цикл с постусловием

Цикл с постусловием используется, как правило, в тех случаях, когда заранее неизвестно число повторений цикла.



Пример использования цикла с постусловием

Задача № 5. Составить программу для вычисления значения функции $y = ax^2$. Переменная x изменяется от 5 до 25 с шагом 1, полученный результат вывести на экран.

Программа решения задачи имеет вид:

```
program post3;                          {Заголовок программы}
  var                                   {Раздел описания переменных}
    x:integer;
    a,y:real;
 begin
   writeln ('Вводи параметр a ');       {Вывод сообщения}
   readln (a);                          {Ввод с клавиатуры значения a}
   y:=0;                                {Обнуление переменной для накапливания
суммы}
   x:=5;                                {Задание начальных условий}
    repeat;                             {Начало  цикла с постусловием}
      y:=a*x*x;                         {Тело цикла с постусловием}
      x:=x+1;                           {Изменение переменной цикла на величину шага}
    until x>25;                         {Проверка условия на выход из цикла}
  writeln ('Функция равна ',y);         {Вывод результата}
end.                                    {Конец программы}
```

Контрольные вопросы и задания

Вопросы:

1. Что такое цикл, управляющая переменная цикла?
2. Оператор цикла с параметром.
3. Оператор цикла с предусловием.
4. Оператор цикла с постусловием.
5. Отличия цикла с параметром от других операторов цикла.
6. Отличия цикла с постусловием от других операторов цикла.

Задания:

1. Составить программу для вычисления суммы ряда:

$$\sum_{n=1}^{45} \frac{1}{n}$$

2. Составить логическую схему алгоритма и для вычисления значения функции: y=2*x*x; при переменной x, изменяющейся от 1 до 3 с шагом 0.1.

3. Составьте программу, которая вычисляет сумму чисел от 1 до N. Значение N (N должно быть меньше 100) вводится с клавиатуры.

4. Напишите программу печати таблицы перевода расстояний из дюймов в сантиметры (1 дюйм = 2,5 см) для значений длин от 1 до 20 дюймов.

5. С помощью while напишите программу вывода всех четных чисел в диапазоне от 2 до 100 включительно.

6. Составьте и отладьте программу, вычисляющую сумму квадратов чисел от 1 до введенного вами целого числа n.

7. С помощью while напишите программу определения суммы всех нечетных чисел в диапазоне от 1 до 99 включительно.

8. С помощью цикла while напишите программу определения идеального веса для взрослых людей по формуле: Ид. вес = рост—100. Выход из цикла: значение роста = 250.

9. С помощью repeat напишите программу-фильтр, которая вводит любые символы, но комментирует только буквы русского алфавита. Завершение работы программы — по нажатии буквы «Я».

10. С помощью repeat напишите программу, которая требует у вас пароль, например 111, и если пароль правильный, то заполняет все строки экрана сообщением «Молодец!!!». Если после пятой попытки пароль все равно неверен, выйти из программы.

11. Составьте и отладьте программу, определяющую максимальное из всех введенных вами чисел. (Пусть признаком конца ввода чисел является введенное число 0.)

2.4. Массивы

При использовании больших объемов данных требуется как-то их структурировать или объединить данные в отдельные группы. Решить такую задачу можно путем использования массивов. **Массив** – это упорядоченная совокупность значений одинакового типа. Например, в программе можно

описать и обрабатывать массивы целых чисел, логических и символьных значений. Массивы могут быть одномерными, двумерными и многомерными.

2.4.1. Одномерный массив

Описание массива

1. VAR < имя >; ARRAY [тип индекса] OF < тип элемента >

Пример:

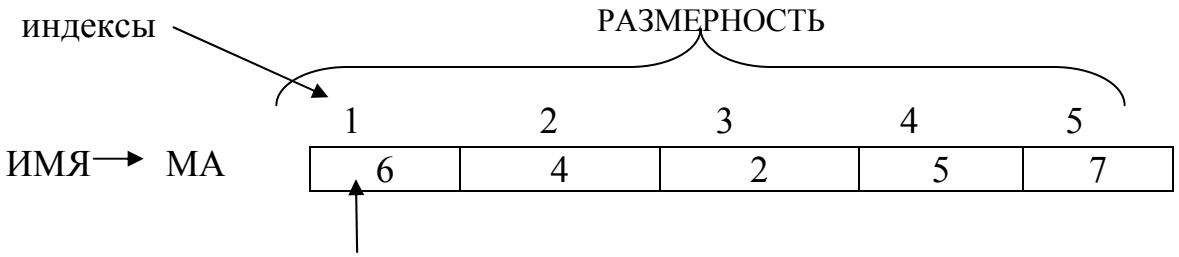
```
VAR
    MA: ARRAY  [1..5] OF INTEGER;
```

2. TYPE < имя типа> = ARRAY [1..5] OF < тип элемента >

VAR < имя массива > < имя типа>

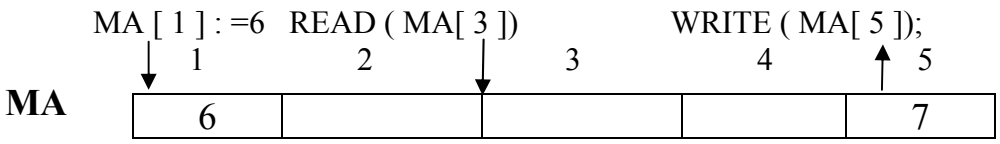
Пример:

```
TYPE M = ARRAY  [1..5] OF INTEGER;
VAR MA:M;
```



ВВОД ЗНАЧЕНИЙ ЭЛЕМЕНТА МАССИВА

ВЫВОД ЗНАЧЕНИЙ ЭЛЕМЕНТА МАССИВА



2.4.2. Двумерный массив

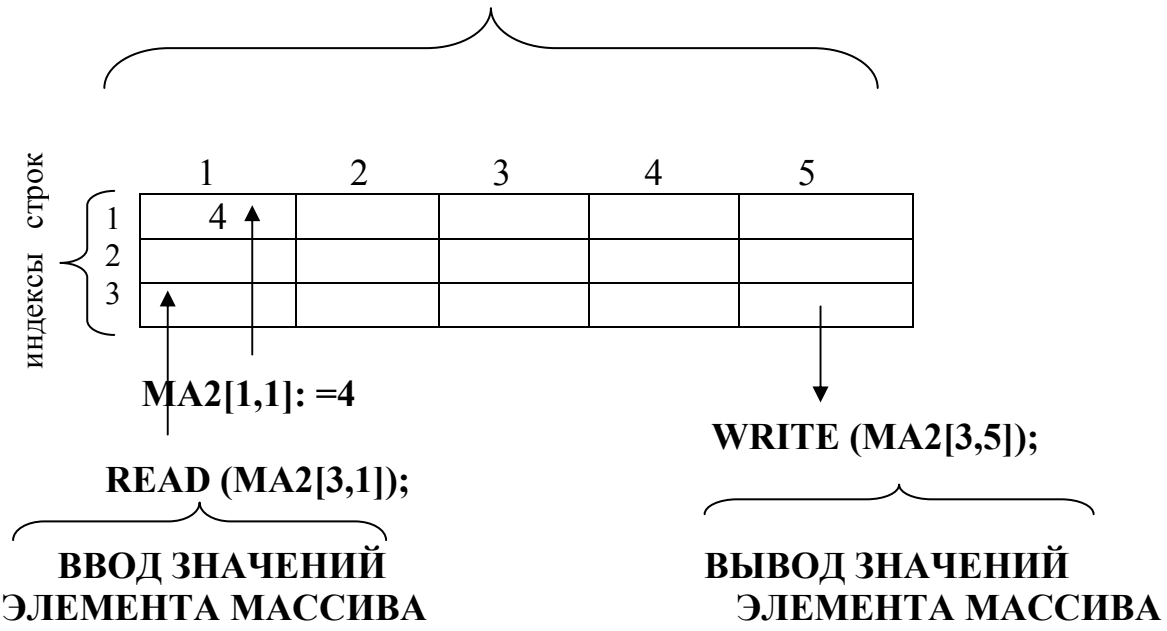
1. VAR < имя >; ARRAY [тип индекса строк, тип индекса столбцов]
< тип элемента >

Пример:

```
VAR
    MA2: ARRAY  [1..3,1..5] OF REAL;
```

2. TYPE M = ARRAY [1..3,1..5] OF REAL;

```
VAR MA2:M:
   . . .
BEGIN
 . . .
READ (MA2[1,3]);
WRITE  (MA2[3,5]);
 . . .      End.
```



Пример использования одномерного массива

Задача № 6. Составить программу вычисления суммы действительных чисел, записанных в одномерный массив a, размером 7 элементов и вывода на экран содержимого введенного массива и полученной суммы.

Программа решения задачи имеет вид:

```
program massiv;                    {заголовок программы}
   const x=7;                       {описание константы}
```

```
 var                                          {Раздел описания переменных}
    s:real;
    i:integer;
    a:array[1..x] of real;                        {описание массива}
begin
   writeln('Введите ',x,' чисел');                    {Вывод сообщения}
          for i:=1 to x do                  {Цикл для ввода элементов массива}
           readln(a[i]);                  {ввод с клавиатуры значений в массив}
   s:=0;                             {Обнуление переменной для накапливания суммы}
        writeln('Введенный массив');
    for i:=1 to x do               {Начало цикла для вывода элементов массива}
           begin
              write(a[i]:5:1,' ');          {вывод значений массива на экран}
              s:=s+a[i];                 {Накапливание суммы элементов массива}
           end;                                    {Конец цикла}
     writeln;                               {переход на новую строку}
     writeln('Сумма элементов массива s=  ',s:5:1);
end.                                                {Конец программы}
```

Пример задачи с двумерным массивом

Задача № 7. Набрать, отредактировать, отладить и выполнить программу формирования единичной матрицы M2(10*10).

Программа решения задачи имеет вид:

```
program mas_2;                                 {заголовок программы}
var
    i,j:integer;
   M2:array[1..10,1..10] of integer;        {описание массива}
begin
        for i:=1 to 10 do                 {Цикл для ввода элементов массива по строкам}
           for j:=1 to 10 do               {Цикл для ввода элементов массива в строке}
             if i=j Then M2[i,j]:=1 Else M2[i,j]:=0; {ввод  значений элементов
массива}
         writeln ('Единичный массив');
        for i:=1 to 10 do          {Цикл для вывода элементов массива по строкам}
           begin                                    {начало цикла по строкам}
             for j:=1 to 10 do{Цикл для вывода элементов массива по элементам
строки}
               write(M2[i,j]:5,' ');              {вывод значений массива строки на
экран}
              Writeln;                                {переход на новую строку}
           End;                                       {конец цикла по строкам}
end.                                                    {Конец программы}
```

Контрольные вопросы и задания

Вопросы:

1. Что такое массив?
2. Как определить местоположение элемента в массиве?
3. Что такое индекс? Каким требованиям он должен удовлетворять?
4. Особенности расположения элементов массива в памяти ЭВМ.
5. Каким образом задается описание массива, что в нем указывается?
6. Общие и отличительные черты одномерных, двумерных и n-мерных массивов.
7. В каких операциях могут участвовать массивы и какие к ним при этом предъявляются требования?
8. Каким образом в **Pascal** задается обращение к элементу массива?

Задания:

1. Введите с клавиатуры в массив X пять целочисленных значений, выведите их в одну строку через запятую; получите для массива среднюю арифметическую.
2. Введите с клавиатуры пять целочисленных элементов массива X. Выведите на экран значения корней и квадратов каждого из элементов массива.
3. Создайте массив из пяти фамилий и выведите их на экран столбиком, начиная с последней.
4. Создайте массив из пяти фамилий и выведите на экран те из них, которые начинаются с определенной буквы, которая вводится с клавиатуры.
5. Дан одномерный массив. Вставьте в него элемент L в позицию K.
6. Введите с клавиатуры целочисленные элементы матрицы 3*3, выведите исходную матрицу на экран. Умножьте каждый элемент матрицы на 3 и выведите результат на экран.
7. Создайте двумерный массив (3*4) целых чисел и найдите сумму всех его элементов.
8. Введите с клавиатуры целочисленные элементы матрицы 3*3 и вычислите сумму элементов каждого столбца.
10. Создайте массив из 15 целочисленных элементов и определите среди них минимальное значение.
11. Создайте двумерный массив X, имеющий четыре строки и три столбца, и найдите в нем максимальный по абсолютному значению элемент, а также укажите номер строки и столбца, содержащие этот элемент. Например, в массиве

```
 2   1   3
-4   0   8
 7   5   1
-3   1   0
```

максимальный по абсолютному значению элемент = 8, находится он во второй строке третьего столбца.

12. Введите массив (не более 20) и определите, есть ли в нем элементы с одинаковыми значениями.

2.5. Подпрограммы

При разработке программ иногда требуется одни и те же последовательности действий выполнять на различных этапах обработки информации. В таких задачах в различных местах встречаются фрагменты, одинаковые по выполняемым действиями, различающихся только в значениях исходных данных. Повторяющаяся группа операторов оформляется в виде самостоятельной программной единицы – подпрограммы. Подпрограмма – это самостоятельная часть программы, реализующая определенный алгоритм и допускающая обращение к ней из различных частей основной программы.

В языке **Pascal** подпрограммы реализуются в виде процедур и функций, которые вводятся в программу с помощью своего описания.

2.5.1. Процедуры

Процедуры описываются в специальном разделе описательной части программы вслед за разделом переменных. Любая процедура состоит, аналогично программе, из заголовка процедуры и тела процедуры.

Заголовок процедуры представляет собой:

PROCEDURE < и м я> (список параметров),

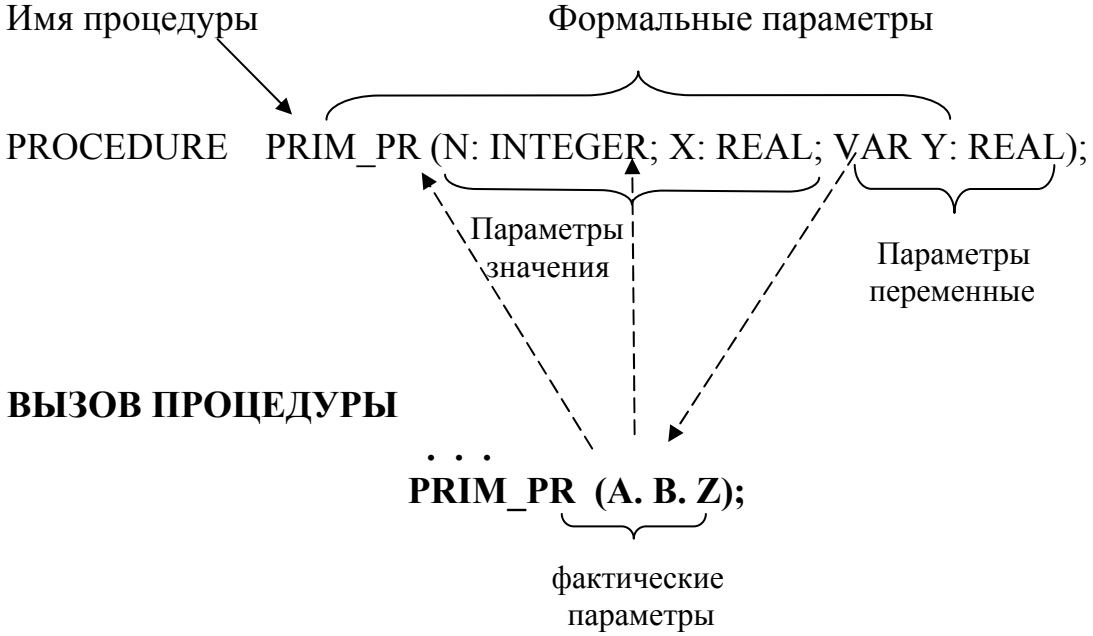
где PROCEDURE – служебное слово;

имя – имя процедуры, определяемое в соответствии с правилами построения идентификаторов;

список параметров – перечень имен для обозначения исходных данных и результатов работы процедуры с указанием их типов.

Пример:

1. ЗАГОЛОВОК ПРОЦЕДУРЫ



Рассмотрим задачу с использованием процедуры

Задача № 8. Составить программу вычисления степени $Z=a^m$, где m – любое целое (положительное или отрицательное) число и ‘а’ не равно нулю. Для решения использовать процедуру с параметром.

$$Z = \begin{cases} a^m, \text{ если } m>0; \\ 1, \text{ если } m=0; \\ 1/(a)^m, \text{ если } m<0. \end{cases}$$

Учитывая, что $1/(a)^{-m} = (1/a)^{-m}$, и используя процедуру с параметром, составим программу.

Программа решения задачи имеет вид:

```
program step;                              {заголовок программы}
   var                                     {описание переменных}
     m:integer;                           {показатель степени}
     a,z:real;                             {число, результат}
   procedure step1 (n:integer; x:real;var y:real); {заголовок процедуры}
     var
      i:integer;
  begin                                   {операторная часть процедуры}
    y:=1;
     for i:=1 to n do                     {цикл для вычисление степени}
       y:=y*x;                             {цикл для вычисление степени}
  end;                                       {конец процедуры}
begin
  writeln (' Введите  a,m ');                {вывод сообщения}
  readln (a,m);                             {ввод с клавиатуры значений в массив}
    If m=0 then z:=1                       {проверка условия, выполнение оператора}
      else if m>0 then step1(m,a,z)  {иначе проверка условия, выполнение
оператора}
            else step1(-m,1/a,z);              {иначе выполнение оператора}
 writeln (a:4:2,' в степени ',m:3,' равно ',z:4:2);  {вывод результата}
 end.                                        {конец программы}
```

2.5.2. Функции

Функция – это подпрограмма, результат выполнения которой есть единственное скалярное значение, присваиваемое имени этой функции. Функция является частным случаем процедур. Отличия процедуры от функции:

- результат выполнения функции – одно значение, а процедуры – одно или несколько;
- результат выполнения функции передается в основную программу как значение имени этой функции, а результаты выполнения процедуры – как значение ее параметров.

Заголовок функции представляет собой:

FUNCTION < и м я> : тип,

где FUNCTION – служебное слово;

имя – имя функции;

тип – тип результата значения, которое должно приобретать имя функции.

ПРИМЕР

1. ЗАГОЛОВОК ФУНКЦИИ

FUNCTION F (N: REAL): REAL;

2. ВЫЗОВ ФУНКЦИИ

PER: = F (K);

Пример решения задачи с использованием функции

Задача № 9. Составьте программу вычисления факториалов Fn=n!, Fm=m!, Fnm=(n-m)!. Вычисление факториала оформить в виде функции с параметрами.

Факториал n! представляет собой произведение n чисел натурального ряда: 1*2*3*...*n.

Программа решения задачи имеет вид:

```
PROGRAM FUNC;
  VAR
   FN,FM,FNM:INTEGER;
   N,M:INTEGER;
   (* ФУНКЦИЯ  FAKT *)
  FUNCTION FACT(K:INTEGER):INTEGER;      {начало описания функции}
   VAR
    P,I:INTEGER;            {Раздел описания  локальных переменных}
   BEGIN                    {начало операторной части функции}
    P:=1;
    FOR I:=1 TO K DO
      P:=P*I;
    FACT:=P;
   END;                     {конец описания функции}
   (* ОСНОВНАЯ ПРОГРАММА *)
BEGIN
   WRITE('ВВЕДИТЕ ЗНАЧЕНИЯ N,M:  ');
   READ(N,M);               {Ввод данных с клавиатуры}
   FN:=FACT(N);             {обращение к функции}
   FM:=FACT(M);             {обращение к функции}
   FNM:=FACT(N-M);          {обращение к функции}
   WRITELN('FN=',FN:5);     {Вывод результата}
   WRITELN('FM=',FM:5);     {Вывод результата}
   WRITELN('FNM=',FNM:5);   {Вывод результата}
END.                        {Конец программы}
```

При использовании подпрограмм без параметров:

1. Глобальные переменные, объявленные в основной части программы, доступны во всех процедурах программы.

2. Локальные переменные, объявленные в подпрограммах, доступны только в данной подпрограмме и внутренних подпрограммах, но не доступны в основной части программы.

Контрольные вопросы и задания

Вопросы:

1. Что называется подпрограммой? В чем состоит сходство и различие подпрограмм-процедур и подпрограмм-функций в языке TURBO PASCAL ?

2. В чем различие между стандартными и определенными пользователем подпрограммами? Приведите примеры.

3. Опишите последовательность событий при вызове процедуры или функции.

4. Что называется параметром, и каково его назначение? Формальные, фактические параметры, их взаимосвязь.

5. Каковы отличия параметров-значений от параметров-переменных, особенности их описания и применения?

6. Каковы особенности параметров-процедур и параметров-функций?

7. Чем отличаются локальные и глобальные параметры? Какова область их действия?

Задания:

1. Напишите программу вычисления выражения
$y=(tg(X)+sin(X))*e^{cos(x)}$ при X=3.7. Результат вывести в формате 5:2.

2. Напишите программу, которая с помощью функции **Chr** выведет на экран кодовую таблицу ПЭВМ (ASCII-таблицу). Задержите выведенную информацию на 5 с и очистите экран.

3. Напишите программу, которая выведет на экран 10 строк по 5 случайных чисел в диапазоне 0..36.

4. Напишите программу, которая по значениям двух катетов вычисляет гипотенузу и площадь треугольника.

5. Напишите функцию возведения в степень по формуле: $A^B = Exp(Ln(A)*B)$
и используйте ее в программе для возведения в 4-ю степень вещественного числа 2,87.

6. Оформите процедуру Proverka пользователя на право работы с программой. Используйте для этого пароль = SCHOO1. Если пароль неправильный, выйти из программы по Halt.

7. Напишите программу, состоящую из трех процедур и основной программы. Первая процедура организует ввод двух целых чисел X и Y, вторая вычисляет их сумму, третья выводит результат. Используйте эти процедуры в основной программе. Используйте X, Y как глобальные переменные. Эта программа послужит прообразом всех ваших будущих программ, т. к. в ней

реализуется принцип работы любой системы: логически выделенные ввод, обработка и вывод результата.

8. Напишите программу вычисления площади поверхности и длины экватора на основе известного радиуса планет солнечной системы. Форму планет будем считать шаром. Вычисление площади и длины экватора оформите отдельными функциями.

9. Составить программу поиска большего из четырех чисел с использованием подпрограммы поиска большего из двух.

10. Даны координаты вершин многоугольника (x1,y1, x2,y2,..., x10,y10). Определить его периметр (вычисление расстояния между вершинами оформить подпрограммой).

11. Вычислить сумму: 1! + 2! + 3! +...+ n!, используя функцию вычисления факториала числа k!

12. Составьте программу вычисления числа сочетаний из N по M. Число сочетаний определяется по формуле N!/(M!*(N—M)!), где N — количество элементов перебора. Используйте подпрограмму вычисления факториала.

13. Определить наименьший общий делитель трех натуральных чисел.

14. Даны действительные числа s,t. Составить программу вычисления выражения

f(t,—2s,1.17) + f(2.2,t, s—t) , где f(a,b,c) = (2a—b—sin(c))/(5+|c|).

15. Составьте программу вычисления суммы квадратов простых чисел, лежащих в интервале (M,N).

2.6. Файлы

Большие совокупности данных, например: телефонный справочник, удобно иметь записанными во внешней памяти в виде последовательности сигналов. В TP для этих целей предусмотрены специальные объекты — файлы. **Файлом** называется совокупность данных, записанная во внешней памяти под определенным именем.

Целесообразность применения файлов диктуется следующими причинами.

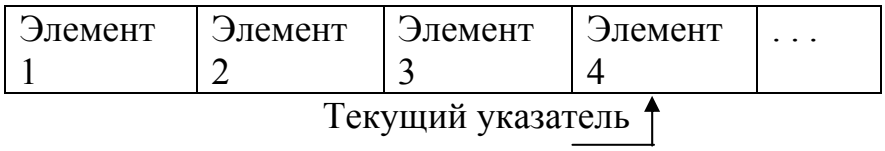
1. Ввод больших объемов данных, подлежащих обработке, утомителен и требует большого времени. Гораздо удобнее создать отдельный файл данных, который может быть подготовлен заранее и, самое главное, применяться неоднократно.

2. Файл данных может быть подготовлен другой программой, становясь, таким образом, связующим звеном между двумя разными задачами, а также средством связи программы с внешней средой.

3. Программа, использующая данные из файла, не требует присутствия пользователя в момент фактического исполнения.

Файл можно представить как потенциально бесконечный список значений одного и того же (базового) типа. Все элементы файла считаются пронумерованными, начальный элемент имеет нулевой номер.

Файл



В любой момент времени программе доступен только один элемент файла, на который ссылается **текущий указатель** (указатель обработки). Часто позицию размещения доступного элемента называют **текущей позицией.**

Как правило, все действия с файлом (чтение из файла, запись в файл) производятся поэлементно, причем в этих действиях участвует тот элемент файла, который обозначается текущим указателем. В результате совершения операций текущий указатель может перемещаться, настраиваясь на тот или иной элемент файла.

По способу доступа к элементам различают файлы последовательного и прямого доступа. **Файлом последовательного доступа** называется файл, к элементам которого обеспечивается доступ в такой же последовательности, в какой они записывались.

Файлом прямого доступа называется файл, доступ к элементам которого осуществляется по адресу элемента. Например, для поиска нужного элемента в последовательном файле необходимо, начиная с нулевого, перемещать указатель обработки до тех пор, пока он не будет указывать на искомый элемент, а при поиске нужного элемента в файле прямого доступа достаточно указать номер его позиции. При организации данных в файл последовательного доступа нельзя одновременно читать данные из файла и записывать данные в файл, так как для чтения некоторого элемента последовательного файла указатель обработки помещен на данный элемент, а для записи нового элемента этот указатель одновременно должен быть в конце файла.

Компилятор TP поддерживает три типа файлов: текстовые, типизированные, нетипизированные.

Средства обработки файлов

При работе с файлами выполняется следующая цепочка команд:

а) при записи файла

```
ASSIGN (F1, TXT, DAN);
REWRITE (F1);
WRITELN (F1, N);
CLOSE (F1);
```

а) при чтении файла

```
ASSIGN (F1, TXT, DAN);
RESET (F1);
WRITELN (F1, N);
CLOSE (F1);
```

F1 – Вспомогательная файловая переменная.

N – Вспомогательная переменная для записи (чтение текста из файла).

Где:

ASSIGN – Логическое подключение из файла TXT, DAN к F1.

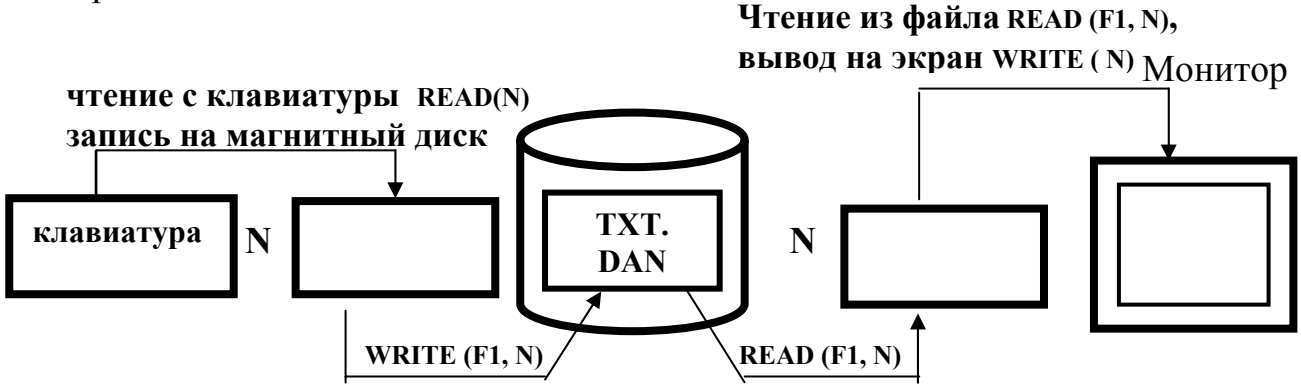
REWRITE (F1) {RESET (F1):} – Открытие файла для записи {чтения} текста.

WRITE (F1, N) {READ (F1, N):} – Запись(чтение) текста в файл на магнитный диск.

CLOSE (F1) – Закрытие файла.

Пример записи данных с клавиатуры во вспомогательную переменную, затем сохранение на магнитный диск показан на левой стороне диаграммы.

Пример чтения данных с магнитного диска во вспомогательную переменную, затем вывод из нее на экран показан на правой стороне диаграммы.



Задача № 10. Постановка задачи: Составить программу формирования файла F, состоящего из целых чисел. Программа показывает работу с файловой переменной F и внешним файлом DAT.TXT. В файл заносятся N= 6 записей, каждая из которых представляет собой целое число.

Программа решения задачи имеет вид:

```
PROGRAM ZAP_TIP;                              {Имя программы}
 VAR                                        {Раздел описания переменных}
   F:FILE OF INTEGER;
   X,I:INTEGER;
 begin
   ASSIGN(F,'DAT.TXT');{подключение файла 'DAT.TXT' к файловой
переменной F}
   REWRITE(F);                              {открытие файла на запись}
   WRITELN('  ВВЕДИТЕ 6 ЦЕЛЫХ ЧИСЕЛ  '); {Вывод сообщения}
 FOR I:=1 TO 6 DO       {цикл для ввода данных}
       BEGIN
         READ(X);                   {чтение с клавиатуры}
         WRITE(F,X);                {запись на магнитный диск в файл}
       END;                         {конец цикла}
   CLOSE(F);                         {закрытие файла}
 END.                                      {Конец программы}
```

Задача № 11. Постановка задачи: Составить программу чтения файла F, состоящего из целых чисел. Программа выполняет работу с файловой переменной F и внешним файлом DAT.TXT. Из файла выводятся целые числа.

Программа решения задачи имеет вид:

```
program CTEN_TIP;                    {Имя программы}
VAR                                  {Раздел описания переменных}
   F:FILE OF INTEGER;
   X,I:INTEGER;
BEGIN                                {начало операторной части программы}
   WRITELN('ЧТЕНИЕ ТИПИЗИРОВАННОГО ФАЙЛА'); {Вывод
сообщения}
ASSIGN(F,'DAT.TXT'); {подключение файла 'DAT.TXT' к файловой
переменной F}
   RESET(F);                                 {открытие файла для чтения}
    WHILE NOT EOF (F) DO    {цикл для чтения данных с магнитного диска на
экран}
       BEGIN
          READ(F,X);              {чтение данных из файла в переменную}
          WRITE(X,' ');           {вывод данных из переменной на экран}
       END;
   CLOSE(F);                      {закрытие файла}
END.                              {Конец программы}
```

Задача № 12. Постановка задачи: Составить программу ввода текстового файла с именем 'TEX.TXT', представляющий собой список необходимой техники. Сделать так, чтобы в каждой строке файла записывалось одно наименование. При вводе каждого наименования начинать с новой строки. На экран вывести из файла 'TEX.TXT' список техники. Каждое наименование вывести с новой строки.

Программа решения задачи имеет вид:

```
PROGRAM VV_TEX;                  {Имя программы}

  VAR                            {Раздел описания переменных}
    SP:TEXT;
     C:CHAR;
    I,N:INTEGER;
BEGIN
  ASSIGN(SP,'TEX.TXT'); {подключение файла 'TEX.TXT' к файловой
переменной F}
  REWRITE(SP);            {открытие файла на запись}
  WRITE('ВВЕДИТЕ КОЛИЧЕСТВО СТРОК В СПИСКЕ:'); {Вывод
сообщения}
  READLN(N);              {вод значений переменной N с клавиатуры}
```

```
 WRITELN('ВВЕДИТЕ НАИМЕНОВАНИЕ ТЕХНИКИ  ',N:1,'
НАИМЕНОВАНИЙ');

FOR I:=1 TO N DO              {цикл для ввода строк текста с клавиатуры в файл}
        BEGIN
          WHILE NOT EOLN DO {цикл для ввода одной строки текста  в файл}
             BEGIN
                READ(C);            {ввод текста в переменную С}
                WRITE(SP,C);        {запись текста из переменной в файл}
             END;                   {конец внутреннего цикла}
          READLN;
          WRITELN(SP);                    {запись в   файл}
        END;                              {конец внешнего цикла}
   CLOSE(SP);                             {закрытие фала}
WRITELN('СПИСОК  ТЕХНИКИ');               {Вывод сообщения}
     RESET(SP);                           {открытие файла на чтение}
        WHILE NOT EOF (SP) DO             {внешний  цикл чтения строк}
          BEGIN
    WHILE NOT EOLN(SP) DO      {Внутренний цикл чтения одной строки}
      BEGIN
        READ(SP,C);                {чтение во вспомогательную переменную}
       WRITE(C)                    {вывод на экран}
      END;                         {конец внутреннего цикла}
     READLN(SP);                   {чтение из  файла}
    WRITELN;                       {переход на новую строку}
          END                      {конец внешнего цикла}
END.                               {конец файла}
```

Контрольные вопросы и задания

Вопросы:

1. Что такое файл?
2. Какие типы файлов применяются в TP?
3. Основные функции для работы с файлами.
4. Основные правила использования файлов в программах.
5. Назовите общие и отличительные черты типизированного и текстового файла.
6. Зачем используется специальная файловая переменная? Как устанавливается соответствие файловой переменной файлу во внешней памяти?
7. Что общего у процедуры **Reset** и **Rewrite** и чем они отличаются?
8. Какие отличия существуют в использовании процедуры **Reset** при открытии различных типов файлов (текстовых, типизированных)?
9. Зачем применяется процедура **Close**?

Задания:

1. Составьте программу, которая создает файл, состоящий из 10 значений типа **integer**. Прочитайте файл и вычислите сумму его элементов.

2. Составьте программу, которая создает файл, состоящий из неопределенного количества значений типа **integer**. Для ввода используйте цикл, выход из цикла — значение 999. После записи выведите файл на экран.

3. Составьте программу, которая создает файл из элементов типа **Char** с помощью цикла **while**. Признак выхода из цикла — буква 'z'. Скопируйте созданный файл в другой файл и выведите его на экран.

4. Составьте программу, которая создает файл, состоящий из пяти значений типа **real**. Выведите файл на экран.

5. Составьте программу, которая создает файл, состоящий из N значений типа **integer**. Прочитайте файл и выведите только четные элементы. Тип **record** не используйте.

6. Составьте программу, которая создает файл из элементов типа **Char** с помощью цикла **while**. Признак выхода из цикла — буква 'z'. Выведите его содержимое на экран.

7. Составьте программу, которая построчно выводит содержимое текстового файла на экран, печатает на бумаге.

2.7. Записи

Для хранения в одном файле данных различного типа в PASCAL применяется комбинированный тип данных – записи.

Запись — это структурированный тип данных, состоящий из фиксированного числа компонентов одного или нескольких типов. Определение типа записи начинается идентификатором **record** и заканчивается зарезервированным словом **end**. Между ними заключен список компонентов, называемых **полями,** с указанием идентификаторов полей и типа каждого поля.

Формат:

```
type
  <имя типа> = record
                <идентификатор поля>:<тип компонента>;
                              ...
               <идентификатор поля>:<тип компонента>
                end;
```

Пример:

```
type
  Car = record
     Number : integer;     {Номер}
     Marka : string[20];   {Марка автомобиля}
     FIO : string[40];     {Фамилия, инициалы владельца}
     Address ; string[60]  {Адрес владельца}
end;
```

В данном примере запись **Car** содержит четыре компонента: номер, название марки машины, фамилию владельца и его адрес. Доступ к полям записи осуществляется через переменную типа «запись». В нашем случае это переменные M и V типа **Car**.

Значения полей записи могут быть использованы в выражениях. Имена отдельных полей не применяются по аналогии с идентификаторами переменных, поскольку может быть несколько записей одинакового типа. **Обращение к значению поля** осуществляется с помощью идентификатора переменной и идентификатора поля, разделенных точкой. Такая комбинация называется **составным именем.** Например, чтобы получить доступ к полям записи **Car**, надо записать:

M.Number, M.Marka, M.FIO, M.Address

Составное имя можно использовать везде, где допустимо применение типа поля. Для присваивания полям значений используется оператор присваивания.

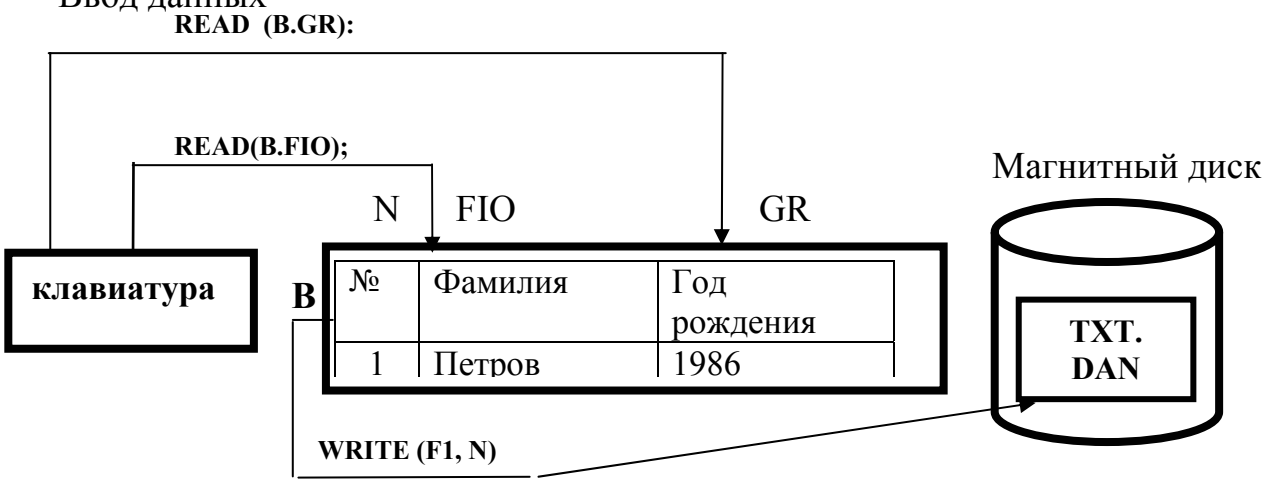
Пример:

M.Number := 1678;

M.Marka := 'ГАЗ - 24';

M.FIO := 'Демьяшкин В.А. ';

Пример ввода записей с клавиатуры в файл и чтение записей из файла на экран схематично показан на диаграмме.



Вывод данных

Магнитный диск

Для ввода и вывода данных используются следующие операторы:

f_zap – вспомогательная файловая переменная;

B – вспомогательная переменная типа записи (для ввода-вывода записей из файла);

Assign (fizap,'sp.doc'); – логическое подключение файла 'spis.dok' на магнитном диске к вспомогательной файловой переменной fizap;

Rewrite(f_zap) - открытие файла на МД с именем 'spis.dok' через вспомогательную файловую переменную fizap для ввода данных;

Reset(f_zap) - Открытие файла на МД с именем 'spis.dok' через вспомогательную файловую переменную fizap для вывода данных;

write(f_zap,b); – запись данных из вспомогательной переменной типа записи b в файл 'spis.dok' на магнитном диске;

read(f_zap,b); - чтение данных из файла 'spis.doc' на магнитном диске во вспомогательную переменную типа записи;

CLOSE (f_zap) – закрытие файла.

Задача № 13. Составить программу ввода списка курсантов группы с указанием фамилии курсанта и года рождения в файл и вывода содержимого файла на экран дисплея.

Исходные данные:

```
-------------------------------------------------------
  № | Фамилия и инициалы   | Год рождения
-------------------------------------------------------
  1  | Иванов С.А.          |  1976
  2  | Андреев П.И.         |  1977
```

Программа решения задачи имеет вид:

```
Program Spis_gr;                    {заголовок программы}
  Uses
    Crt;
  Type
```

```
      zap=record                                  {Описание структуры записи}
        fio:STRING[10];                           {поле фамилия на 10 символов}
        gr:integer;                               {поле год рождения}
      end;
  Var
    f_zap:file of zap;                            {файловая переменная}
    b:zap;                                        {переменная типа записи}
    i,n:integer;                                  {переменные для циклов}
  Begin                                         {начало операторной части программы}
     Assign(f_zap,'sp.doc'); {подключение файловой переменной к файлу 'sp.doc'}
     Rewrite(f_zap);                        {открытие файла с именем  'sp.doc' на
запись}
     Write('Введите кол-во курсантов');              {вывод сообщения}
     Readln(n);                                    {ввод количества строк}
     Write('Введите фам и год рожд курсантов');    {вывод сообщения}
      For i:=1 to n do                              {цикл для ввода данных}
        begin
          write('=>');                            {вывод сообщения}
          read(b.fio);                              {ввод фамилии}
          readln(b.gr);                             {ввод года рождения}
          write(f_zap,b);  {сохранение записи в файле 'sp.doc' на магнитном диске}
        end;                                        {конец цикла}
    Close(f_zap);                                  {закрытие файла}
    {ВЫВОД ИЗ ФАЙЛА}
    I:=1;                                       {установка значения счетчика на начало}
    WRITELN;                                    {указатель на начало новой строки}
    WRITELN('      СПИСОК КУРСАНТОВ');              {вывод сообщения}
    WRITELN('------------------------------');      {вывод сообщения}
    WRITELN(' № |   ФАМИЛИЯ   | ГОД РОЖДЕНИЯ');  {вывод сообщения}
    WRITELN('------------------------------');          {вывод сообщения}
    Reset(f_zap);                                  {открытие файла на чтение записи}
      While not eof(f_zap) do      {цикл для чтения записей из файла и вывода  на
экран}
        begin
          read(f_zap,b); {чтение записи с магнитного диска в переменную типа
записи}
          writeln(I:2,' |', b.fio:10,'   |    ',b.gr);          {вывод полей записи на экран}
          I:=I+1;        {увеличение значения счетчика для вывода номера строки}
        end;                                                    {конец цикла}
    WRITELN('------------------------------');                {вывод сообщения}
    Writeln('конец ');                                        {вывод сообщения}
End.                                                          {конец программы}
```

Контрольные вопросы и задания

Вопросы:

1. Почему запись называют комбинированным типом данных?
2. Как определяется тип записи? Что называется полем записи?
3. Какие требования предъявляются к идентификаторам поля в записи?
4. Чем определяется объем памяти, требуемый для размещения записи?
5. Что такое составное имя поля записи? Из каких частей оно состоит и как записывается?
6. Зачем при обращении к полю записи используется предложение with?
7. Как вы понимаете вложение записей? Каков максимально допустимый уровень вложения? Приведите примеры вложения записей.

Задания:

1. Опишите запись с именем типа **Karta**, содержащую следующие поля:

• номер измерения (тип **integer**);

• значение (тип **real**).

Переменную, определяющую запись, назовите Z.

2. Опишите запись с именем типа **Doc**, содержащую следующие поля:

• номер строки документа (тип **integer**);

• текст строки (тип **string**).

Переменную, определяющую запись, назовите S.

3. Опишите запись с именем texn, содержащую информацию о хранящемся на складе техники:

• код техники (тип **integer**);

• наименование техники (тип **string**);

• цену (тип **real**).

Переменную, определяющую запись, назовите Tov.

4. Опишите запись с именем типа **Graf**, содержащую данные, необходимые для построения графика из 40 точек:

• название графика (тип **string**);

• 40 значений (тип **integer**).

Переменную, определяющую запись, назовите X.

5. Опишите запись с именем типа Bata, содержащую информацию для базы данных:

• личный номер обучаемого (тип **integer**);

• ФИО (тип **string**);

• год рождения (тип **integer**);

• адрес (тип **string**).

Переменную, определяющую запись, назовите Inf.

6. Опишите запись с именем типа Systema, содержащую информацию о планетах солнечной системы:

• номер планеты по удалению от Солнца (тип **integer**);

• название планеты (тип **string**);

• объем (**real**);

• диаметр (**real**);

• удаленность от Земли (**real**).

Переменную, определяющую запись, назовите Planeta.

7. Опишите запись с именем типа Sport, содержащую информацию о лучших спортивных достижениях школы по легкой атлетике:

• название вида (тип **string**);

• фамилия рекордсмена (тип **string**);

• дата установления рекорда (запись Dat, состоящая из полей Day, Month, Year);

• сообщение о результате (**real**).

Переменную, определяющую запись, назовите Rec.

8. Опишите запись с именем типа Geometr, содержащую информацию об оценках студентов вашей группы по ОБЖ:

• ФИО (тип **string**);

• оценки за девять месяцев max по 20 оценок в месяц.

Переменную, определяющую запись, назовите Dig.

9. Опишите запись с именем типа Rasp, содержащую информацию о движении электропоездов из вашего города:

• направление (тип **string**);

• время отправления электропоездов (тип **real**).

Переменную, определяющую запись, назовите R.

3. СПРАВОЧНАЯ СИСТЕМА TURBO PASCAL

Интегрированная среда программирования **Turbo Pascal** отличается расширенными возможностями встроенной справочной системы, которая позволяет программисту не только получить контекстно-ориентированную справочную информацию, но и делать вырезки и вставки кода примеров для каждой библиотечной процедуры и функции в текст своей программы, возвратиться назад к другим экранам подсказки (клавиши [Alt + Fl]), воспользоваться подсказкой по справочной информации (клавиша [F1], если вы уже находитесь в системе справочной информации).

Примечание. Название «контекстно-ориентированная» справочная система **Turbo Pascal** получила за возможность получения справочной информации, связанной с текущим состоянием среды программирования, по указанному элементу языка программирования. Например, для получения справочной информации о любом пункте меню интегрированной среды программирования активизируйте этот пункт и нажмите клавишу [F1]; для получения справки по элементу языка в окне редактирования (оператору, функции и т.п.) установите курсор на нужном элементе и нажмите клавиши [Ctrl + F1].

Для получения справочной информации (за исключением случаев, когда управление переходит к вашей программе) нужно нажать клавишу [F1] или

отметить мышью нужный пункт меню **Help**. Меню **Help** (клавиша [Alt+H]) обеспечивает вас таблицей содержания системы справочной информации, подробным оглавлением, способностями поиска [Ctrl + F1]. Любой экран справочной информации может содержать одно ключевое слово или несколько ключевых слов (высвеченных элементов), по которым можно получить дополнительную справочную информацию.

Использование клавиш для получения справочной информации отражает таблица.

Клавиша (и)	Элемент меню	Функция
F1	Help/Contents	Открывает контекстно-ориентированный экран справочной информации
F1, F1	Help/Help on Help	Вызывает справочную информацию по справочной информации (нужно нажать только клавишу [F1], если вы уже находитесь в системе справочной информации)
Shift + F1	Help/Index	Вызывает оглавление справочной информации
Alt + F1	Help/Previous Topic	Показывает предыдущий экран справочной информации
Ctrl + F1	Help/Topic Search	Вызывает специфическую информацию по языку только в редакторе

ПОЛУЧЕНИЕ СПРАВОЧНОЙ ИНФОРМАЦИИ ПО РЕДАКТОРУ

Для получения справочной информации по операциям редактирования клавишей [F1] вызовите экран подсказки; нажимая клавишу [Page Down], перейдите к перечню подсказок о функциях редактирования.

ЗАКЛЮЧЕНИЕ

Существует большое количество языков алгоритмического программирования и, если понятен принцип работы, имеются навыки практической работы одного из них, освоить другой обычно несложно.

Рассмотренный в данном пособии материал позволяет достаточно твердо освоить один из популярнейших языков программирования **Turbo Pascal**. Язык программирования **Turbo Pascal** последних версий имеет многооконный режим работы, возможность создавать объектно-ориентированные программы, возможность использовать типизированный адресный оператор, а так же улучшенный интерфейс для пользователя.

Автор не ставил перед собой цель рассмотреть полностью все аспекты программирования и управления данными, а старался представить материал в легко усваиваемом и доступном для восприятия стиле, с целью получения первых практических навыков по программированию на языке **Turbo Pascal**.

Направлением дальнейшей работы автора является разработка 2 и 3 частей пособия в которых более подробно будут рассмотрены стандартные библиотечные модули языка **Turbo Pascal** , управление динамическими данными и основы объектно-ориентированного программирования.

Приложение 1

Сообщение компилятора об ошибках

Если ошибка возникает при компиляции внутри TP, **Turbo Pascal** активизирует окно редактирования и помещает курсор на место ошибки в исходной программе.

Если ошибка возникает при использовании командно-строчного компилятора, **Turbo Pascal** выдает сообщение об ошибке, выводит на экран исходную строку программы с ошибкой и ее номер; символ ^ в выдаваемой исходной строке указывает местоположение ошибки. Ошибки подразделяются на:

- ошибки при компилировании программы;
- ошибки на уровне DOS;
- ошибки ввода-вывода.

Ошибки при компилировании

1 Out of memory (Выход за пределы памяти)

Данная ошибка появляется, если компилятор израсходовал всю доступную ему память. Возможно, ваша программа или программный модуль слишком велики, чтобы компилировать их в таком объеме памяти. В этом случае программу или программный модуль необходимо разбить на два или более программных модуля.

2 Identifier expected (Ожидается идентификатор)

В этом месте должен находиться идентификатор.

3 Unknown Identifier (Неизвестный идентификатор)

Этот идентификатор не был описан.

4 Duplicate Identifier (Повторение идентификатора)

5 Syntax error (Синтаксическая ошибка)

В исходном тексте найден недопустимый символ. Возможно, не заключена в кавычки строковая константа.

6 Error in real constant (Ошибка в константе вещественного типа)

Ошибка в синтаксисе константы вещественного типа.

7 Error In Integer constant (Ошибка в константе целого типа)

Ошибка в синтаксисе константы целого типа. (Учтите, что после чисел, превышающих диапазон представления целых чисел, должны ставиться точка и нуль, например 12345678912.0.)

8 String constant exceeds line (Строковая константа превышает размеры строки)

Вероятно, после строковой константы отсутствует символ кавычка.

9 Too many nested files (Слишком много вложенных файлов)

Компилятор допускает не более 15 вложенных исходных файлов и не более 4 включаемых файлов.

10 Unexpected end of file (Неожиданный конец файла)

Данное сообщение может появиться по одной из следующих причин:

В Вашей программе, вероятнее всего, неодинаковое количество операторов BEGIN и END.

Включаемый файл заканчивается в середине раздела операторов. (Каждый раздел операторов должен целиком помещаться в одном файле)

В Вашем файле не закрыты скобки комментария.

11 Line too long (Строка слишком длинная)

Максимальная длина строки не может превышать 127 символов.

12 Type Identifier expected (Ожидается идентификатор типа)

В определенном месте не указан тип идентификатора.

13 Too many open files (Слишком много открытых файлов)

В файле CONFIG.SYS не установлен параметр FILES = N,

где N — представляет собой целое число без знака.

Установите в файле CONFIG.SYS значение параметра FILES = N >= 30.

14 Invalid file name (Недопустимое имя файла)

Имя файла неверно или указан несуществующий путь.

15 File not found (Файл не найден)

Файл не найден ни в одном из каталогов, предназначенных для хранения файлов данного типа.

16 Disk full (Диск заполнен)

Удалите с диска ненужные Вам файлы (например, файлы с расширением .BAK или .TMP) или попробуйте сохранить текст программы на другом диске, который имеет достаточный объем свободной памяти.

17 Invalid compiler directive (Недопустимая директива компилятора)

Неверная буква в директиве компилятора, один из параметров директивы компилятора неверный или Вы пользуетесь глобальной директивой компилятора, когда компиляция тела программы уже началась.

18 Too many files (Слишком много файлов)

В компиляции программы или программного модуля участвует слишком много файлов. Попытайтесь не использовать так много файлов, например, объединяя включаемые файлы.

19 Undefined type a pointer definition (Неопределенный тип в описании указателя)

Вы пытаетесь объявить типизированный указатель, связанный с ранее необъявленным типом данных.

20 Variable Identifier expected (Ожидается идентификатор переменной)

В указанном курсором месте ожидается идентификатор переменной.

21 Error in type (Ошибка определении типа)

Определение типа не может начинаться с этого символа.

22 Structure too large (Слишком длинная структура)

Максимально допустимый размер любого структурированного типа — 65520 байтов.

23 Set base type of range (Количество элементов в множестве превышает допустимое значение)

Базовый тип множества должен представлять собой интервальный или перечисляемый тип данных с не более чем 256 значениями.

24 File components may not be files or objects (Компоненты файла не могут быть файлами или объектами)

Тип компонентов файла не может быть объектным типом, файловым типом или любым другим структурированным типом, содержащим компоненты типа файла или объекта.

25 Invalid string length (Неверная длина строки)

Длина описываемой строки должна находиться в пределах от 1 до 255.

26 Type mismatch (Несоответствие типов)

Причины, вызвавшие появление данного сообщения, могут быть следующими:

Несовместимы типы переменной и выражения в операторе присваивания.

Несовместимые типы фактического и формального параметров в обращении к процедуре или функции.

Тип выражения несовместимый с типом индекса при индексировании массива.

Несовместимые типы операндов в выражении.

27 Invalid subrange base type (Неправильный базовый тип для интервала)

28 Lower bound greater than upper bound (Нижняя граница превышает верхнюю)

При описании интервального типа данных Вы объявили нижнюю границу диапазона больше верхней.

29 Ordinal type expected (Ожидается перечисляемый тип)

Вещественные, строковые, структурные и ссылочные типы в данном случае не допускаются.

30 Integer constant expected (Ожидается константа целого типа)

31 Constant expected (Ожидается константа)

32 Integer or real constant expected (Ожидается константа целого или вещественного типа)

33 Pointer type Identifier expected (Ожидается идентификатор типа указатель)

Указанный курсором идентификатор не является указателем.

34 Invalid function result type (Недопустимый тип результата функция)

Правильными типами результата функции являются все простые, строковые и ссылочные типы.

35 Label Identifier expected (Ожидается идентификатор метки)

Обнаружена ссылка на метку, не описанную в разделе LABEL.

36 BEGIN expected (Ожидается оператор BEGIN)

37 END expected (Ожидается оператор END)

38 Integer expression expected (Ожидается выражение целого типа)

39 Ordinal expression expected (Ожидается выражение перечисляемого типа)

40 Boolean expression expected (Ожидается выражение логического типа)

41 Operand types do not match operator (Несоответствие типов операнду)

Данная операция не может применяться к операндам этого типа. Такое сообщение будет вызвано, например, при попытке выполнить следующую операцию:' 9' DIV 'G' .

42 Error in expression (Ошибка в выражении)

Данный идентификатор не может участвовать в выражении указанным образом.

Возможно, вы забыли указать операцию между двумя операндами.

43 Illegal assignment (Запрещенное присваивание)

Это сообщение может появиться по следующим причинам:

– файлам и нетипизированным переменным нельзя присваивать значения;

– идентификатору функции можно присваивать значения только внутри раздела операторов данной функции.

44 Field Identifier expected (Ожидается идентификатор поля)

Данный идентификатор не соответствует полю предшествующей переменной типа RECORD или OBJECT.

45 Object file too large (Объектный файл слишком большой)

Turbo Pascal не может компоновать OBJ-файлы размером больше чем 64 Кбайта.

46 Undefined external (Не определена внешняя процедура)

Внешняя процедура или функция не имеет соответствующего определения PUBLIC в объектном файле. Убедитесь, что Вы указали все объектные файлы в директивах {$L <имя OBJ-файла>), и проверьте написание идентификаторов процедуры или функции в файле .ASM.

47 Invalid object file record (Недопустимая запись объектного файла)

Файл .OBJ содержит неверную объектную запись. Убедитесь в том, что данный файл является действительно OBJ-файлом.

48 Code segment too large (Сегмент кода слишком большой)

Максимально допустимый размер кода программы или программного модуля ранен 65520 байтам. Разбейте Вашу программу или программный модуль на несколько частей.

49 Data segment too large (Сегмент данных слишком большой)

Максимальный размер сегмента данных программы равен 65520 байтам, включая данные, описываемые используемыми программными модулями. Если нужно большее количество глобальных данных, опишите большие структуры с помощью указателей и выделяйте для них память динамически с помощью процедуры New.

50 DO expected (Ожидается оператор DO)

51 Invalid PUBLIC definition (Недопустимое определение PUBLIC)

Появление этого сообщения возможно по следующим причинам:

– Данный идентификатор получил тип PUBLIC с помощью соответствующей директивы языка ассемблер, но не соответствует описанию EXTERNAL в программе или программном модуле.

– Две или более директивы PUBLIC на языке ассемблера определяют один и тот же идентификатор.

– OBJ-файл определяет идентификатор PUBLIC вне сегмента CODE.

52 Invalid EXTRN definition (Неправильное определение EXTRN)

Появление этого сообщения возможно по следующим причинам:

- Из Ассемблера была осуществлена ссылка с помощью директивы EXTRN на идентификатор, который не был описан в тексте Pascal -программы.
- Идентификатор обозначает абсолютную переменную.
- Идентификатор обозначает процедуру или функцию типа INLINE.

53 Too many EXTRN definition (Слишком много определений типа EXTRN)

Turbo Pascal не может обрабатывать файлы .OBJ при более чем 256 определениях EXTRN.

54 OF expected (Ожидается оператор OF)

55 INTERFACE expected (Ожидается оператор INTERFACE)

56 Invalid relocatable reference (Недопустимая перемещаемая ссылка)

Появление этого сообщения возможно по следующим причинам:

– OBJ-файл содержит данные и перемещаемые ссылки в сегментах, отличных от CODE, например, при попытке описать инициализированные переменные в сегменте DATA.

– ORJ-файл содержит ссылки с размерами в байтах на перемещаемые символы. Такая ошибка происходит в случае использования операторов HIGH и DOWN с перемещаемыми символами или, если Вы ссылаетесь в директивах DB на перемещаемые символы.

– Операнд ссылается на перемещаемый символ, который не был определен в сегментах CODE или DATA.

– Операнд ссылается на процедуру EXTRN или функцию EXTRN со смещением, например CALL SortProc+8.

57 THEN expected (Ожидается оператор THEN)

58 TO or DOWNTO expected (Ожидается зарезервированное слово TO или DOWNTO)

59 Undefined forward (Неопределенное опережающее описание)

Появление этого сообщения возможно по следующим причинам:

– Была описана процедура или функция в интерфейсной секции программного модуля, но их определение отсутствует в секции реализации.

– Процедуры или функции были описаны с помощью опережающего описания, но их определение не найдено.

61 Invalid typecast (Недопустимое преобразование типов)

Размер переменной и тип результата отличаются друг от друга при приведении типа переменной.

Вы пытаетесь осуществить приведение типа выражения, когда разрешается только ссылка на переменную, процедуру или функцию.

62 Division by zero (Деление на нуль)

Предшествующая операция пытается выполнить деление на нуль.

63 Invalid file type (Недопустимый тип файлов)

Данный файловый тип не обслуживается процедурой обработки файлов. Например, процедура ReadLn используется для типизированного файла или процедура Seek — для текстового файла.

64 Cannot Read or Write variables of this type (Нет возможности считать или записать переменные данного типа)

Эта ошибка может появиться из-за попытки ввести или вывести переменную несоответствующего типа:

– Процедуры Read и ReadLn могут считывать переменные символьного, целого, вещественного и строкового типов.

– Процедуры Write и WriteLn могут выводить переменные символьного, целого, действительного, булевого и строкового типов.

65 Pointer variable expected (Ожидается переменная типа указатель)

Предыдущая переменная должна иметь тип указатель.

66 String variable expected (Ожидается строковая переменная)

Предшествующая переменная должна иметь строковый тип.

67 String expression expected (Ожидается выражение типа строка)

Предшествующее выражение должно иметь строковый тип.

68 Circular unit reference (Циклическая зависимость модулей)

69 Unit name mismatch (Несоответствие имен программных модулей)

Имя программного модуля, найденное в файле .TPU, не соответствует имени, указанному в операторе USES.

70 Unit version mismatch (Несоответствие версий программных модулей)

Один или несколько программных модулей, используемых данной программой, были изменены после их компиляции. Воспользуйтесь командой Compile/Make (Компиляция/Формирование) или Compile/Build (Компиляция/Построение) в интегрированной интерактивной среде программирования и параметрами/M или/B в компиляторе TP, что позволит автоматически скомпилировать программные модули, нуждающиеся в перекомпиляции.

71 Duplicate unit name (Повторное имя программного модуля)

Имя этого программного модуля уже указано в операторе USES.

72 Unit file format error (Ошибка формата файла программного модуля)

TPU-файл является недействительным.

Убедитесь, что это действительно TPU-файл соответствующей версии языка.

73 Implementation expected (Ожидается оператор IMPLEMENTATION)

В модуле отсутствует раздел реализации.

74 Constant and case types do not match (Несовпадение типов константы и оператора CASE)

Тип константы оператора CASE несовместим с выражением в операторе варианта.

75 Record variable expected (Нужна переменная типа запись)

Предшествующая переменная должна иметь тип запись.

76 Constant out of range (Константа вне диапазона)

Эта ошибка может появиться по следующим причинам:

– При попытке указать массив с константами, нарушающими границы.

– При попытке присвоить переменной значение константы, выходящее за диапазон переменной.

– При попытке передать константу вне диапазона в качестве параметра процедуре или функции.

77 File variable expected (Ожидается файловая переменная)

Предшествующая переменная должна иметь файловый тип.

78 Pointer expression expected (Ожидается выражение типа указатель)

Предшествующее выражение должно иметь тип указателя.

79 Integer or real expression expected (Ожидается выражение целого или вещественного типа)

Предшествующее выражение должно иметь тип INTEGER или REAL.

80 Label not within current block (Метка вне пределов текущего блока)

Оператор GOTO не может осуществить переход на метку, находящуюся вне текущего блока.

81 Label already defined (Метка уже определена)

Данная метка уже помечает точку перехода.

82 Undefined label in processing statement part (Неопределенная метка в предыдущей части оператора)

83 Invalid @ argument (Недействительный аргумент оператора @)

Правильными аргументами являются имена переменных, процедур или функций.

84 Unit expected (Ожидается оператор UNIT)

85 ";" expected (Ожидается символ ";")

86 ":" expected (Ожидается символ ":")

87 "," expected (Ожидается символ ",")

88 "(" expected (Ожидается символ "(")

89 ")" expected (Ожидается символ ")")

90 "=" expected (Ожидается символ "=")

91 ":=" expected (Ожидается символ ":=")

92 "[" or "(." expected (Ожидается символ "[" или "(.")

93 "]" or ".)" expected (Ожидается символ "]" или ".)")

94 "." expected (Ожидается символ ".")

95 ".." expected (Ожидается символ "..")

96 Too many variables (Слишком много переменных)

Эта ошибка может появиться по следующим причинам:

– Объем памяти, занимаемый всеми описанными в программе или программном модуле глобальными переменными, не может превышать 64 Кбайтов.

– Объем памяти, занимаемый описанными в программе или функции локальными переменными, не может превышать 64 Кбайтов.

97 Invalid FOR control variable (Недопустимая переменная управления циклом FOR)

Параметр цикла оператора FOR должен быть переменной перечисляемого типа.

98 Integer variable expected (Ожидается переменная целого типа)

Предшествующая переменная должна иметь целый тип.

99 File: are not allowed here (Файлы и типы процедур здесь не разрешены)

Типизированная константа не может иметь файловый тип.

100 String length mismatch (Несовпадение длины строки)

Длина строковой константы не соответствует количеству элементов символьного массива.

101 Invalid ordering of fields (Недопустимый порядок полей)

Поля в константе типа RECORD должны записываться в порядке их описания.

102 String constant expected (Ожидается константа строкового типа)

103 Integer or real variable expected (Ожидается переметая целого или вещественного типа)

Предшествующая переменная должна иметь целый или вещественный тип.

104 Ordinal variable expected (Ожидается переменная перечисляемого типа)

Предшествующая переменная должна иметь перечисляемый тип.

105 INLINE error (Ошибка в операторе INLINE)

Оператор < не допускается в сочетании с перемещаемыми ссылками на переменные. Такие ссылки всегда имеют размер в слово.

106 Character expression expected (Ожидается выражение символьного типа)

108 Overflow in arithmetic operation (Переполнение при выполнении математических операции)

Значение результата последней выполненной математической операции превышает допустимые размеры типа Longlnt (-2147483648 ... 2147483647). В этом случае рекомендуется использовать Overflow in arithmetic operation.

109 No enclosing FOR, WHILE or REPEAT statement (Не найдены операторы цикла)

Эта ошибка возникает в том случае, когда стандартные процедуры Break и Continue используются вне операторов цикла FOR, WHILE или REPEAT.

112 CASE constant out of range (Константа в операторе CASE вне диапазона допустимых границ)

Значение целочисленных констант оператора CASE должно находиться в пределе от -32768 до 32767.

113 Error In statement (Ошибка в операторе)

Символ, на который указывает курсор, не может быть первым символом в операторе.

114 Cannot call an interrupt procedure (Невозможен вызов процедуры обработки прерывания)

Вы не можете непосредственно вызвать процедуру прерывания.

116 Must be in 8087 mode to compile this (Для компиляции необходим режим 8087)

Данная конструкция может компилироваться только в режиме {$N+}. В состоянии {$N-} операции с типами REAL (одиночной и двойной точности, расширенными и сложными) не допускаются.

117 Target address not found (Указанный адрес не найден)

Команда Search/Find Error (Компиляция/Поиск ошибки) в интегрированной интерактивной среде или опция /F в командной строке компилятора не позволяют обнаружить оператор, соответствующий указанному адресу.

118 Include files are not allowed here (В данном месте программы подключение файла невозможно)

Раздел операторов должен целиком размещаться в одном файле.

119 No inherited methods are accessible here (Недопустимое использование наследуемых методов)

Ошибка возникает в том случае, если зарезервированное слово INHERITED используется вне метода объектного типа или внутри метода объектного типа, не имеющего предков.

120 NIL expected (Ожидается оператор NIL)

Типизированные константы или указатели могут инициализироваться только значением NIL.

121 Invalid qualifier (Неверный квалификатор)

Эта ошибка может появиться по следующим причинам:

– При попытке индексировать переменную, которая не является массивом.

– При попытке указать поля в переменной, которая не является записью.

– При попытке применить оператор * к переменной, которая не является указателем.

122 Invalid variable reference (Недопустимая ссылка на переменную)

Предыдущая конструкция удовлетворяет синтаксису ссылки на переменную, но она не указывает адрес памяти. Наиболее вероятно, что вызвана функция-указатель, но отсутствует ссылка (с помощью символа") на результат.

123 Too many symbols (Слишком много символов)

Программа или программный модуль описывает более 64 Кбайтов символов. Если Вы компилируете программу с помощью директивы {$D+}, то попробуйте отключить эту директиву или попытайтесь разбить программу на несколько модулей.

124 Statement part too large (Слишком большой раздел операторов)

Turbo Pascal ограничивает размер раздела операторов до величины примерно 24 Кбайтов. Если вы обнаружили эту ошибку, поместите части разделов операторов в одну или несколько процедур. В любом случае при наличии раздела операторов такого размера не стоит жалеть усилий, чтобы сделать более ясной и понятной структуру своей программы.

126 Files must be var parameters (Файлы должны иметь переменные в качестве параметров)

Попытка передать процедуре или функции параметр-значение файлового типа. Укажите ключевое слово VAR перед переменными.

127 Too many conditional symbols (Слишком много символов в условном выражении)

Отсутствует место для определения условных символов. Попытайтесь удалить некоторые идентификаторы или сократить некоторые из ключевых имен компиляции.

128 Misplaced conditional directive (Пропущена условная директива)

Компилятор обнаружил директиву {$ELSE) или {$ENDIF} без соответствующих директив ($IFDEF), ($IFNDEF) или ($IFOFT).

129 ENDIF directive mlssing (Пропущена директива ENDIF)

Исходный файл закончился внутри конструкции условной компиляции. В исходном файле должно быть равное количество директив и {$ENDIF}.

130 Error in Initial conditional defines (Ошибка в определениях начальных условных выражений)

Исходные условные идентификаторы, указанные в опции Options/Compller/Condltional Defines (Параметры/Компилятор/Условные определения) или в параметре /D компилятора командной строки, являются недопустимыми.

131 Header does not match previous definition (Заголовок не соответствует предыдущему определению)

Эта ошибка может появиться по следующим причинам:

– Заголовок процедуры или функции, указанный в интерфейсной секции, не соответствует заголовку исполняемой части процедуры или функции.

– Заголовок процедуры или функции, указанный с помощью опережающего описания FORWARD, не соответствует заголовку найденной одноименной процедуры или функции.

132 Critical disk error (Критическая ошибка диска)

Во время компиляции произошла критическая ошибка диска (например, дисковод находился в состоянии неготовности).

133 Cannot evaluate this expression (Невозможно вычислить данное выражение)

В выражении-константе или в отладочном выражении используются неподдерживаемые средства, например, в описании константы используется функция Sin или в отладочном выражении вызывается определенная пользователем функция.

136 Invalid Indirect reference (Недопустимый косвенный указатель)

Предшествующий оператор пытается осуществить недопустимую косвенную ссылку. Например, используется абсолютная переменная, базовая переменная которой в текущем модуле неизвестна, или в программе типа INLINE используется ссылка на переменную, не определенную в текущем модуле.

137 Structured variable are not allowed here (В данном месте использование структурной переменной не допускается)

Предпринята попытка выполнения над структурной переменной неподдерживаемой операции. Например, попытка перемножить две записи.

140 Invalid floating-point operation (Недопустимая операция с вещественным числом)

При операции с двумя действительными значениями было получено переполнение или деление на нуль.

142 Procedure or function variable expected (Ожидается процедура или функция)

В этом контексте оператор получения адреса @ может использоваться только с переменной-процедурой или функцией.

143 Invalid procedure or function reference (Недопустимые указатель па процедуру или функцию)

Эта ошибка может появиться по следующим причинам:

– Вы пытались вызвать процедуру в выражении.

– Если конкретную реализацию процедуры или функции нужно присвоить переменной-процедуре, то она должна компилироваться с использованием ключа {$F+} и не может описываться с помощью ключевых слов INLINE или INTERRUPT.

146 File access denied (Оказано в доступе к файлу)

Файл не может быть открыт или создан. Скорее всего, компилятор пытается произвести запись в файл с атрибутом Read only (Только для чтения).

147 Object type expected (Ожидается объектный тип)

Идентификатор не определяет объектный тип или данный тип упущен.

148 Local object types not allowed (Локальные объектные типы не разрешены)

Объектные типы могут быть определены только в глобальном блоке программы или модуля. Объявление типа объекта внутри процедур (функций) и модулей не допускается.

149 VIRTUAL expected (Ожидается VIRTUAL)

В описании объекта отсутствует ключевое слово VIRTUAL.

150 Method Identifier expected (Ожидается идентификатор метода)

Указанный идентификатор не является идентификатором метода.

151 Virtual constructor are not allowed (Конструктор нельзя объявлять виртуальным)

Правило конструктора должно быть статическим.

152 Constructor identifier expected (Ожидается идентификатор конструктора)

Данный идентификатор не является конструктором объекта.

153 Destructor Identifier expected (Ожидается идентификатор деструктора)

Данный идентификатор не является идентификатором деструктора.

154 Fall only allowed within constructors (Вызов FAIL допускается только внутри конструктора)

Стандартная процедура FAIL может быть вызвана только из конструктора объекта.

155 Invalid combination of opcode and operands (Недопустимая комбинация кода операции и операндов)

Код ассемблерной команды не воспринимает данное сочетание операндов. Появление этого сообщения возможно по следующим причинам:

– Внутри операторов ассемблера использованы комментарии, например MOV {начальное значение}AX,1.

– Указано слишком много или слишком мало операндов для данной команды, например INC AX,BX или MOV AX.

– Количество операндов правильно, но их тип и порядок не соответствуют коду операции, например DEC I, MOV AX,CL или MOV 1,AX.

156 Memory *reference expected (Ожидается ссылка на область памяти)*

Операнд ассемблерной инструкции не является требуемым указателем на область памяти. Скорее всего, в указании индексных регистров операнда отсутствуют квадратные скобки, например MOV AX.BX+SI вместо MOV AX.[BX+S1).

157 Cannot add or substract relocatable symbols (Сложение или вычитание перемещаемых символов невозможно)

С перемещаемыми идентификаторами в операнде Ассемблера допускается выполнение единственной операции — это сложение с константой или вычитание константы. Переменные, процедуры, функции и метки представляют собой перемещаемые идентификаторы. Предположим, что Var — это переменная, а Const — константа. Тогда инструкции MOV AX, Const+Const и MOV AX.Var+Const являются допустимыми, а MOV AX,Var+Var — нет.

158 Invalid register combination (Недопустимая регистровая комбинация)

Допустимыми комбинациями индексных регистров являются [BX], [BP], [SI], [DI], [BX+SI], [BX+DI], [BP+SI] и [BP+DI]. Другие комбинации индексных регистров, например [AX], [BP+BX] и [S1+DX], не допускаются. Заметим, что локальные переменные (переменные, описанные в процедуре или функции) размещаются в стеке и доступ к ним организуется через регистр BP. При ссылках на такие переменные Ассемблер автоматически добавляет [BP], поэтому, хотя конструкция

типа Local[BX] (где Local — локальная переменная) и выглядит допустимой, операндом в итоге будет Local[BP+BX].

159 286/287 Instructions not allowed (Инструкции процессоров 286/287 не разрешены)

Используйте директиву компилятора {$G+}, но имейте в виду, что результирующий код не сможет работать на машинах с процессорами 6086 и 8088.

160 Invalid symbol reference (Недопустимая ссылка на идентификатор)

Данный идентификатор в операнде Ассемблера недоступен. Эта ошибка может появиться по следующим причинам:

– Вы пытались обратиться к стандартной процедуре, стандартной функции или специальным массивам Mem, MemW, MemL, Port, PortW.

– Вы обратились к строковой, вещественной константе в операторе ассемблерной команды.

– В операнде Ассемблера Вы пытались обратиться к процедуре или функции типа INLINE.

– Вы пытались получить с помощью операции @Result доступ к результату, возвращаемому функцией.

– Вы пытались использовать короткую инструкцию команды JMP, которая выполняет переход не на метку, а на что-то другое.

161 Code generation error (Ошибка генерации кода)

Ошибка возникает, в частности, при компиляции ассемблерных фрагментов, содержащих команды LOOPNE, LOOPE, LOOP или JCXZ, если команда ссылается на недоступную метку.

162 ASM expected (Ожидается ключевое слово ASM)

163 Duplicate dynamic method Index (Дублирование индекса динамического метода)

Индекс динамического метода уже использован другим динамическим методом. Возможно, вы пытались переопределить динамический метод, но ошиблись в имени, введя таким образом новый метод.

Ошибки на уровне DOS

1 Invalid function number (Ошибочный номер функции)

Обращение к несуществующей функции DOS.

2 File not found (Не найден файл)

Ошибка генерируется процедурами Reset, Append, Rename или Erase, если физический файл, связанный с файловой переменной, не найден или не существует.

3 Path not found (Путь не найден)

Ошибка генерируется процедурами Reset, Append, Rename или Erase, если имя присвоенное файловой переменной является недействительным или

указывает на несуществующий подкаталог. Ошибка генерируется процедурами ChDir, MkDir или RmDir, если маршрут является недействительным или указывает на несуществующий подкаталог.

4 Too many open files (Слишком много открытых файлов)

Ошибка генерируется процедурами Reset, Rewrite или Append, если программа имеет слишком много открытых файлов.

5 File access denied (Отказано в доступе к файлу)

Появление этого сообщения возможно по следующим причинам:

– Данная ошибка генерируется процедурой Reset или Append, если переменная FileMode допускает запись, в то время как физический файл является каталогом или файлом, доступным только для чтения.

– Данная ошибка генерируется процедурой Rewrite, если каталог заполнен, или если имя, присвоенное файловой переменной, задает каталог или существующий файл, доступный только для чтения.

6 Invalid file handle (Недопустимый описатель файла)

Данная ошибка генерируется, если системному вызову DOS передается недопустимый описатель файла. Появление данной ошибки является свидетельством того, что файловая переменная испорчена.

12 Invalid file access code (Неверный код доступа к файлам)

Ошибка генерируется процедурами Reset или Append при попытке открыть файл (типизированный или нетипизированный), если значение переменной FileMode в момент открытия файла было недействительным.

15 Invalid drive number (Недопустимый номер диска)

Ошибка генерируется процедурой GetDir или ChDir, если номер диска недопустим.

16 Cannot remove current directory (Нельзя удалить текущий каталог)

Ошибка генерируется процедурой RmDir при попытке удалить текущий каталог.

17 Cannot rename across drives (Нельзя при переименовании указывать разные диски)

Ошибка генерируется процедурой Rename, если оба файла не находятся на одном и том же диске.

18 No more files (Больше нет файлов)

Эта ошибка передается в переменную DOSError модулей DOS и WinDos, если при вызове процедур FindFirst и FindNext не найдено файлов с заданным именем и набором атрибутов.

Ошибки ввода-вывода

100 Disk read error (Ошибка чтения диска)

Ошибка генерируется процедурой Read при попытке осуществить считывание после конца типизированного файла.

101 Disk write error (Ошибка записи на диск)

Ошибка генерируется процедурами Close, Write, WriteLn, Flush, если на диске нет свободного места.

102 File not assigned (Файл не связан)

Ошибка генерируется процедурами Reset, Rewrite, Append, Rename и Erase, если с файловой переменной не было связано имя физического файла посредством обращения к процедуре Assing.

103 File not open (Файл не открыт)

Ошибка генерируется процедурами Close, Read, Write, Seek, Eof, FilePos, FileSIze, Flush, BlockRead или BlockWrite при попытке осуществить операции ввода-вывода с файлом, который еще не открыт.

104 File not open for Input (Файл не открыт для ввода)

Ошибка генерируется процедурами Read, ReadLn, Eof, Eoln, SeekEof или SeekEoln, если текстовый файл не открыт для чтения.

105 File not open for output (Файл не открыт для вывода)

Ошибка генерируется процедурами Write, WriteLn, если текстовый файл не открыт для записи.

106 Invalid numeric format (Недопустимый числовой формат)

Ошибка генерируется процедурами Read или ReadLn, если числовое значение, считанное из текстового файла, не соответствует числовому формату соответствующего типа данных.

Критические ошибки

150 Disk is write-protected (Диск защищен от записи)

151 Bad drive request structure length (Неправильная длина структуры запроса дисковода)

152 Drive not ready (Дисковод не готов)

153 Unknown command (Неизвестная команда)

154 CRC error In data (Ошибка контроля данных)

155 Bad drive request structure length (При обращении к диску указана неверная длина структуры)

156 Disk seek error (Ошибка при поиске дорожки диска)

157 Unknown media type (Неизвестный тип носителя)

158 Sector not found (Сектор не найден)

159 Printer out of paper (Принтер без бумаги)

160 Device write fault (Неисправное устройство записи)

161 Device read fault (Неисправное устройство чтения)

162 Hardware failure (Сбой аппаратных средств).

Приложение 2

НАЗНАЧЕНИЕ ФУНКЦИОНАЛЬНЫХ КЛАВИШ

Горячая клавиша	Эквивалентная команда меню	Функция
[F1]	–	Активизация окна контекстно-зависимой помощи
[Alt+F1]	–	Возврат к предыдущей справке
[Ctrl+F1]	–	Активизация синтаксической справки, т. е. справки об операторе, на который указывает маркер
[Shift+F1]	Help/Index	Вызов содержания справочной подсистемы
[F2]	[F1]	—
[Ctr+F2]	[Alt+F1]	Help/Previous topic
[F3]	[Ctrl+F1]	Help/Topic search
[Alt+F3]	Window /Close	Закрытие активного окна
[Ctrl+F3]	Debug/Call Stack	Открытие окна протокола используемых процедур
[F4]	Run/ Goto Cursor	Выполнение программы, расположенной в активном окне, до позиции курсора
[Ctrl+F4]	Debug/Evaluate/Modif y	Присмотр и изменение значений переменных
[F5]	Window/Zoom	Изменение (увелич./уменьшен.) размера активного окна
[Alt+F5]	Debug/User Screen	Переключение на пользовательский экран
[Ctrl+F5]	Window/Size/ Move	Изменение положения и размера окна
[F6]	Window/Next	Переход к следующему окну
[Snift+F6]	Window/Pfevious	Возврат к предыдущему окну
[F7]	Run/Trace Into	Трассировка программы пооператорно с пооператорным выполнением всех подпрограмм
[Alt+F7]	Tools/Goto previous	Переход к предыдущей строке в окне сообщений
[Ctrl+F7]	Debug/Add watch	Дополнение списка переменных, наблюдаемых в Watch-окне
[F8]	Run/Step over	Трассировка программы пооператорно с выполнением подпрограмм без пооператорной детализации
[Alt+F8]	Tools/Go to next	Переход к следующей строке в окне сообщений
[Ctrl+F8]		Установка /отмена контрольной точки на строке программы, указываемой курсором

Окончание приложения 2

Горячая клавиша	Эквивалентная команда меню	Функция
[F9]	Compile/ Make	Компиляция и редактирование связей программы
[Alt+F9]	Compile/ Compile	Компиляция программы из активного окна
[Ctrl+F9]	Run/Run	Компиляция и выполнение программы под управлением интегрированной инструментальной оболочки
[F10]	—	Активизация строки меню
[Alt+F10]	—	Вызов локального меню
[Alt+Литера]		Открытие озаглавленного выбранной литерой подчиненного меню из строки меню
[Alt+Backsp]	Edit/Undo	Отмена всех изменений в текущей строке
[Alt+X]	File/Exit	Завершение сеанса работы с ТР с сохранением (после подтверждения) файлов, измененных редактором текста
[Alt+Цифра]	—	Переход к окну с указанным номером
[Alt+0]	Window/List	Вызов окна, в котором содержится список всех открытых окон
[Ctrl+Del]	Edit/Clear	Удаление выделенного блока
[Ctrl+Ins]	Edit/Copy	Копирование блока в буфер промежуточного хранения
[Shift+Del]	Edit/Cut	Перенос выделенного блока в буфер промежуточного хранения
[Shift+Ins]	Edit/Paste	Копирование блока из буфера промежуточного хранения в окно редактирования

Приложение 3

Зарезервированные слова

Зарезервированные слова являются составной частью языка, имеют фиксированное начертание и раз и навсегда определенный смысл. Они не могут изменяться программистом. Зарезервированные слова версии языка PASCAL для персональных ЭВМ приведены ниже.

Зарезервированные слова версии языка Паскаль для ПЭВМ

absolute	абсолютный	**label**	метка
and	логическое И	**library**	библиотека
array	массив	**mod**	остаток от деления
asm	ассемблер	**nil**	отсутствие
begin	начало блока	**not**	логическое НЕ
case	вариант	**or**	логическое ИЛИ
const	константа	**of**	из
constructor	констуктор	**object**	объект
div	деление нацело	**packed**	упакованный
goto	переход на	**procedure**	процедура
do	выполнять	**program**	программа
downto	уменьшить до	**record**	запись
destructor	деструктор (разрушитель)	**repeat**	повторять
else	иначе	**set**	множество
end	конец блока	**shi**	сдвиг битов влево
exports	экспорт	**shr**	сдвиг битов вправо
external	внешний	**string**	строка
file	файл	**then**	то
for	для	**to**	увеличивая
forward	опережающий	**type**	тип
function	функция	**unit**	модуль
if	если	**until**	до
implementation	реализация	**uses**	использовать
in	в (входит в...)	**var**	переменная
inline	основной	**while**	пока
interrupt	прерывание	**with**	с
interface	интерфейс	**xor**	исключающее ИЛИ
inherited	наследование		

Приложение 4

СТАНДАРТНЫЕ БИБЛИОТЕЧНЫЕ МОДУЛИ

В систему Turbo Pascal версии 6.0 и старше включены восемь модулей:

System, Crt, Dos, Graph, Graph3, Overlay, Printer, Турбо «3» и специализированная библиотека Turbo Vision. Модуль System подключается по умолчанию, все остальные должен подключать программист с помощью зарезервированного слова uses. Например: uses Crt, Dos, Printer.

Рассмотрим кратко назначение каждого из модулей.

System — сердце Turbo Pascal ; содержащиеся в нем подпрограммы обеспечивают работу всех остальных модулей системы.

Crt — содержит средства управления дисплеем и клавиатурой компьютера.

Dos — включает средства, позволяющие реализовывать различные функции DOS.

Graph3 — поддерживает использование стандартных графических подпрограмм версии Turbo Pascal 3.0.

Overlay — содержит средства организации оверлейных программ.

Printer — обеспечивает быстрый доступ к печатающему устройству.

Turbo «3» — обеспечивает максимально возможную совместимость с версией Turbo Pascal 3.0.

Graph — содержит пакет графических средств, обеспечивающих эффективную работу с адаптерами CGA, EGA, VGA, HERC, IBM 3270, MCGA и ATT6300.

Turbo Vision — библиотека объектно-ориентированных подпрограмм для разработки пользовательских интерфейсов.

Встроенные функции и процедуры

Модуль System подключается к программе автоматически, поэтому его имя не указывается в разделе uses. По этой причине программе становятся доступны его встроенные процедуры и функции.

Арифметические процедуры и функции

Abs(X:real/integer):real/integer — вычисление абсолютной величины X. Тип результата совпадает с типом параметра.

ArcTan(X:real):real — вычисление угла, тангенс которого равен X радиан.

Cos(X:real):real — вычисление косинуса X; параметр задает значение угла в радианах.

Exp(X:real):real — вычисление экспоненты X, т. е. значение E в степени X.

E является основанием натурального логарифма и равно 2.718282.

Frac(X:real):real — вычисление дробной части X.

Int(X:real):real — вычисление целой части X.

Ln(X:real):real — вычисление натурального логарифма **X,** т. е. логарифма по основанию e (e = 2.718282).

Pi:real — возвращает значение числа Пи (3.141592653897932385).

Sin(X:real):real — вычисление синуса X. Параметр задает значение угла в радианах.

Sqr(X) — возведение в квадрат значения целого или вещественного значения X. Тип результата совпадает с типом параметра.

Sqrt(X:real):real — вычисление квадратного корня из X.

Random:real — генерирует значение случайного числа из диапазона 0..0.99.

Random(I:word):word — генерирует значение случайного числа из диапазона 0..I.

Randomize — изменение базы генератора случайных чисел.

Скалярные процедуры и функции

Dec(X{,n}) — процедура уменьшает значение целочисленной переменной X на величину n. При отсутствии необязательного параметра n значение X уменьшается на единицу.

Inc(X{,n}) — процедура увеличивает значение целочисленной переменной на n. При отсутствии необязательного параметра n значение X увеличивается на единицу.

Pred(S) — функция возвращает элемент, предшествующий S в списке значений типа. Тип результата совпадает с типом параметра. Если предшествующего S элемента не существует, возникает программное прерывание.

Succ(S) — функция возвращает значение, следующее за S в списке значений типа. Тип результата совпадает с типом параметра. Если следующее за S значение отсутствует, возникает программное прерывание.

Odd(I:integer):boolean — возвращает True, если I нечетное, и False, если I четное.

Функции преобразования типов

Chr(I:byte):char — возвращает символ стандартного кода обмена информацией с номером, равным значению I. Если значение параметра больше 255, возникает программное прерывание.

Ord(S):longint — возвращает порядковый номер значения S в множестве, определенном типом S.

Round(X:real):longint — возвращает значение X, округленное до ближайшего целого числа.

Trunc(X:real):longint — возвращает ближайшее целое число, меньшее или равное X, если X >= 0, и большее или равное X, если X< 0.

Процедуры управления программой

Delay(Lword) — задержка выполнения программы на I мс.

Exit — выход из выполняемого блока в окружающую среду. Если текущий блок является процедурой или функцией, выход производится во внешний блок. Если Exit указана в операторной части основной программы, программа прекращает работу, и управление передается системе программирования.

Halt(N:word) — прекращение выполнения программы и передача управления системе программирования (если выполнялся .PAS-файл) или DOS (если выполнялся .EXE-файл). N — код завершения программы, передаваемый в операционную систему.

RunError(ErrCode:word) — прекращение выполнения программы и генерация ошибки времени выполнения. Err-Code — параметр типа byte, содержащий номер ошибки.

Специальные процедуры и функции

FillChar(P,Dl,Z) — заполняет побайтно область основной памяти заданным значением (заполнителем). Является одной из самых быстродействующих процедур. Область начинается с первого байта указанной переменной P и имеет размер, заданный параметром Dl. P — переменная любого типа;

Dl — целочисленное выражение, указывающее длину; Z — заполнитель, выражение литерного или байтового типа.

Move(P1,P2,Dl) — пересылает содержимое основной памяти, начиная с первого байта переменной P1, в область, которая начинается с первого байта переменной P2. Длина областей определяется параметром Dl.

P1 и P2 — переменные любого типа;

Dl — целочисленное выражение.

Hi(I:integer):byte — выделяет старший байт значения I и помещает его в младший байт результата. Старший байт результата равен 0.

Lo(I:integer):byte — выделяет младший байт значения I и помещает его в младший байт результата. Старший байт результата равен 0.

ParamCount : string — возвращает число параметров, переданных программе в командной строке.

ParamStr (n:word) : string — возвращает указанный параметр командной строки.

SizeOf(IT):word — вычисляет объем основной памяти в байтах, которую занимает указанная переменная или тип. IT — идентификатор переменной или типа данных.

Swap(I:integer):integer — обменивает содержимое младшего и старшего байтов целочисленного выражения, заданного параметром I типа integer.

Вызов стандартной процедуры или функции. Ранее мы уже рассматривали примеры программ, в которых использовались некоторые стандартные процедуры и функции. Для использования стандартной процедуры или функции к программе подключается тот или иной специализированный библиотечный модуль, в котором записана данная стандартная процедура или функция (исключение составляет модуль System, так как он подключается к программе автоматически), для чего имя специализированного библиотечного

модуля указывается в разделе uses. Затем в программе записывается вызов процедуры или функции, для чего записывается ее имя и указываются фактические параметры, например: Pi, Sin(X), Chr(125), Inc(X,5). Так как после выполнения функции ее значение присваивается имени, то имя функции используется в выражении.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Основы информатики: Учебное пособие./ В. З. Аладьев, Ю.Я. Хунт, М.Л. Шишаков. - М.: Информационно-издательский дом «Филин», 1998.
2. Программирование в среде TURBO PASCAL -7.0. А. И. Марченко, Л. И. Марченко. - Киев: ВЕК, 1998.
3. Информатика: Учебное пособие./ А. В. Могилев. – М.:Academa, 2000.
4. TURBO PASCAL. С. А. Немнюгин. - С-Пб.: Питер, 2000.
5. TURBO PASCAL: Учебное пособие./ В. Б. Попов. - М.: Финансы и статистика, 2003 г.
6. Информатика. Базовый курс: Учебное пособие./ С. В. Симонович. - С-Пб.: Питер, 2000.
7. TURBO PASCAL -7.0. А. А. Стеценко. - Киев: BHV, 1998.

Учебное издание

КРАСНОВ Сергей Васильевич

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ
TURBO PASCAL

Учебное пособие

Редактор М. В. Теленкова

Подписано в печать 30.10.2004. Формат 60 × 84/16. Бумага тип. №1. Печать трафаретная. Усл. печ. л. 4,42. Уч.-изд. л. 4,00. Тираж 100 экз. Заказ .

Ульяновский государственный технический университет
432027, г. Ульяновск, ул. Сев. Венец, д. 32
Типография УлГТУ, 432027, г. Ульяновск, ул. Сев. Венец, д. 32