МОУ ДПО «Институт повышения квалификации»
г. Новокузнецк

А.А. Киселева

Основы программирования в Turbo Pascal

Учебное пособие

Новокузнецк, 2005

Учебное пособие посвящено традиционной методике программирования на языке Паскаль, и его реализации в системе программирования Turbo Pascal.

Теоретические сведения и листинги программ с подробными комментариями позволят освоить данный язык программирования за минимальное время.

Учебное пособие предназначено для слушателей курса «Программирование», обучаемых по дополнительной профессиональной образовательной программе профессиональной переподготовки «Теория и методика обучения информатики».

Обзор языков программирования

Файл с командами (исполнимый файл) – и есть программа. **Программа** – упорядоченный список команд. **Инструменты программирования –** инструментальные команды, созданные с помощью других инструментальных более примитивных программ. Прослеживая историю создания одних команд с помощью других, непременно мы дойдем до того момента, когда никаких программ в распоряжении программистов не было, а имели они дело непосредственно с процессором, понимающим только определенный числовой код. Это код называется **машинным** кодом. Каждый процессор понимает свой собственный машинный код. Совокупность кодов, которые понимает и исполняет процессор, называется **системой команд**. Из этих команд и состоят компьютерные программы. Инструменты программирования – специальные программы, которые читают то, что написал программист в удобном ему виде, и переводят его записи в машинный код, необходимый процессору. Но иногда программисту приходиться какую-то часть программы записывать на машинном коде, понятном процессору.

Девяносто пять процентов программы состоит из небольших стандартных подпрограмм (**процедур**). Существуют библиотечные файлы, из которых извлекают стандартные блоки и используют без какой-либо переделки. Год от года такие библиотеки наращиваются. Поэтому с каждым годом растет производительность труда программистов. Библиотеки процедур бывают стандартными, коммерческими, фирменными.

Программы пишутся с помощью специальных **языков программирования**. Языки программирования понятны программисту, но непонятны процессору. Процессор может работать только с числами, и потому понимает только программы, написанные в машинном коде. Поэтому программы сначала «переводят» на язык процессора. Этот перевод выполняют специальные программы-переводчики (**трансляторы**).

Существуют специальные программы-отладчики машинного кода. Они позволяют «заглянуть» в машинный код программы во время ее работы. Кроме того, они позволяют узнать, что находиться в регистрах процессора в какой-то момент времени, даже позволяет вносить изменения в машинный код работающих программ. Отладчик **debug.exe** (c:\windows\command\debug.exe) служебная программа MS-DOS. У отладчика своя система команд, которые могут быть введены с клавиатуры (например: q – выход из программы). Чтобы просмотреть коды программ, зашитых в ПЗУ (BIOS), надо в командной строке набрать: d F000:0000, и нажать Enter. Программа в машинном коде – это набор байтов, процессор сам разбирается, какие байты выражают команды какие – числа, какие –

символы, а какие – адреса в памяти компьютера. Машинный код можно понять, если использовать специальный код, который называется **кодом ассемблера**. По сути это тот же машинный код, но записанный в виде **мнемоник**. Мнемоник ассемблера столько – же, сколько команд в системе команд процессора. Чтобы просмотреть на машинный код, записанный в мнемониках: в командной строке отладчика нужно набрать с клавиатуры u F000:0000, и нажать Enter (add – сложить, exchg – обменять).

Программа на языке программирования записывается с помощью более понятных человеку слов и символов. Языки близкие к процессору называются языками **низкого уровня**, а языки удобные для людей – **высокого уровня**. Язык самого низкого уровня – это язык машинного кода. Чуть выше лежит язык ассемблера, далее идут сотни всевозможных прочих языков. Программу на языке программирования записывают в обычном текстовом редакторе. Такой программный код, записанный на языке программирования высокого уровня, называют **исходным модулем (кодом, текстом)**. Исходный текст программы состоит из специальных команд – **операторов**. Процессор их исполнить не может, и исходный код преобразуется в инструкции процессора. Это преобразование берет на себя **транслятор**. Есть 2 вида трансляторов: **компиляторы и интерпретаторы**. **Компилятор** преобразует исходный код в машинный. В результате образуется так называемый **объектный модуль**. Он записан в машинном коде, но работать пока он не может – к нему надо подключить стандартные процедуры, которые использовал программист. Эти процедуры выбираются из библиотек, прилагающихся к языку программирования. Операцию выбора процедур из библиотек и подключения их к объектному коду выполняет специальная программа – **редактор связей**. Только после этого получается работающая программа – **рабочий код или исполняемый код**. **Интерпретаторы** – трансляторы, которые обрабатывают текст не заранее, а непосредственно во время работы программы.

Скорость работы программы почти всегда сказывается на ее качестве. Откомпилированные программы работают в 20-50 раз быстрее, чем программы, выполняемые под управлением интерпретатора. Интерпретатор выполняет роль посредника между программой и процессором и забирает себе большую часть ресурсов компьютера. К тому же он много раз повторяет одни и те же операции. Поэтому практически все прикладные и служебные программы поставляются в откомпилированном виде. Файлы таких программ имеют расширение имени .EXE или .COM. Это чистый машинный код. Языки программирования, для которых существуют программы-компиляторы, называют **компилируемыми** (Паскаль, C++, Delphi, Fortran…).

Интерпретируемые языки часто используются в качестве учебных (Basic).

Языки программирования делят на: **процедурные** (Fortran, Pascal, Basic, C), **логические** (Лисп, Пролог) и **объектно-ориентированные** (C++, JAVA).

Таблица 1. Классификация языков программирования

Категория	Представление программ и данных	Представление о работе программы	Связь частей программы между собой
Процедурные	Программа и обрабатываемые данные представляют собой отдельные, не связанные друг с другом элементы	Работа программы рассматривается как последовательное выполнение операторов	Связь различных частей программы осуществляется только через данные, которые могут обрабатываться ими совместно.
Объектно-ориентированные	Данные и методы их обработки инкапсулированы в рамках единого объекта	Работа программы рассматривается как последовательность событий и соответствующих реакций различных объектов на эти события	Отдельные части программы могут наследовать методы и элементы данных друг у друга
Логические	Данные и правила их обработки объединены в рамках единого логического структурного образования	Работа программы рассматривается как преобразование этого образования в соответствии со строгими логическими правилами	Разбиение программы на отдельные независимые части затруднительно

Изучение любого языка программирования строится по следующей схеме:

Рисунок 1. Схема изучение языка программирования

Основы программирования в Turbo Pascal

1. Некоторые сведения о системе TP

- Чтобы начать работу над новой программой, необходимо войти в меню редактора, нажав на *F10* после чего выбрать пункт *File* и команду *New*.
- Для сохранения программы на диске: войти в меню редактора, выбрать пункт меню *File*, и команду *Save* в нем, в появившемся окне указать путь и имя файла, нажать клавишу *Enter*, файлу автоматически будет присвоено расширение .PAS.
- Для открытия файла: *File* ⇨ *Open*, в появившемся окне указать имя диска, на котором располагается файл, установить маркер на имени файла и нажать клавишу *Enter*.
- Чтобы откомпилировать программу: нужно нажать комбинацию клавиш *Alt+F9*.
- Для выполнения откомпилированной программы: *Ctrl+F9*.
- Для просмотра результатов работы программы в текстовом режиме: *Alt+F5*.
- Справка: *F1* – получение контекстно-зависимой справки, *Ctrl+F1* – получение справки о нужной стандартной процедуре, функции, константе или переменной.

2. Структура программы.

По определению стандартного Паскаля программа состоит из следующих блоков:

Название программы начинается со слова **Program,** после которого записывается через пробел название программы.

Блок подключения дополнительных модулей начинается слова **USES,** после которого указываются названия модулей: Graph (содержит функции, позволяющие использовать графические возможности ПК), Crt (функции для опроса клавиатуры, управления курсором …).

Блок описания констант и переменных. Описание начинается со слова **CONST**, затем конкретным константам дается имя, знак «=», значение. Описание переменных начинается со слова **VAR**, затем указывается к какому типу принадлежит переменная, ставится двоеточие и имя переменной. Вот некоторые типы данных: Integer – целочисленные данные, 2 байта, от –32768 до +32767; Real – вещественные данные, 6 байт, от 2.9E-39 до 1.7E+38, точность 11…12 значащих цифр; Char – символ, 1 байт; String – строка символов, занимает максимальное число символов в строке +1; Boolean – логический тип, занимает 1 байт и имеет 2 значения FALSE, TRUE.

Блок описания процедур и функций. Процедура самостоятельный законченный элемент программы, который может быть вызван и выполнен либо в процедуре, либо в блоке выполнения действия программы.

```
PROCEDURE <имя процедуры>;
VAR <описание переменных, использующиеся в процедуре>;
BEGIN
  <Тело процедуры>
END;
```

Блок выполнения действий начинается со слова **BEGIN**, затем идет тело программы, заканчивается словом **END** с точкой на конце.

Пример 1. Первая программа

```
Program P1;                    {название программы}
Const Text = 'Привет';         {описание константы Text}
Begin                          {тело программы}
 Writeln (Text);
End.
```

3. Ввод с клавиатуры и вывод на экран

Ввод данных – это передача информации от внешних устройств в оперативную память. Вводятся, как правило, исходные данные решаемой задачи. Вывод данных – обратный процесс, когда данные передаются из оперативной памяти на внешние носители (монитор).

Процедура ввода с клавиатуры имеет следующий формат:

Read(ln) (<список ввода>)

где <список ввода> - это последовательность имен переменных, разделенных запятыми. Оператор Readln отличается от Read тем, что после считывания последнего в списке значения для одного оператора Readln данные для следующего оператора будут считываться с начала новой строки.

Оператор вывода на экран (обращение к стандартной процедуре вывода) имеет следующий формат:

Write(ln) (<список ввода>)

Пример 2. Создать программу, в которой сначала запрашивалось бы Ваше имя, а потом выводилось на экран: 'Привет', имя.

```
program p1;
var name:string;
begin
  writeln ('Введите Ваше имя');
  read(name);
  writeln('Привет ',name,'!');
end.
```

4. Типы данных

Структура данных, практически во всех процедурных языках имеет следующий вид:

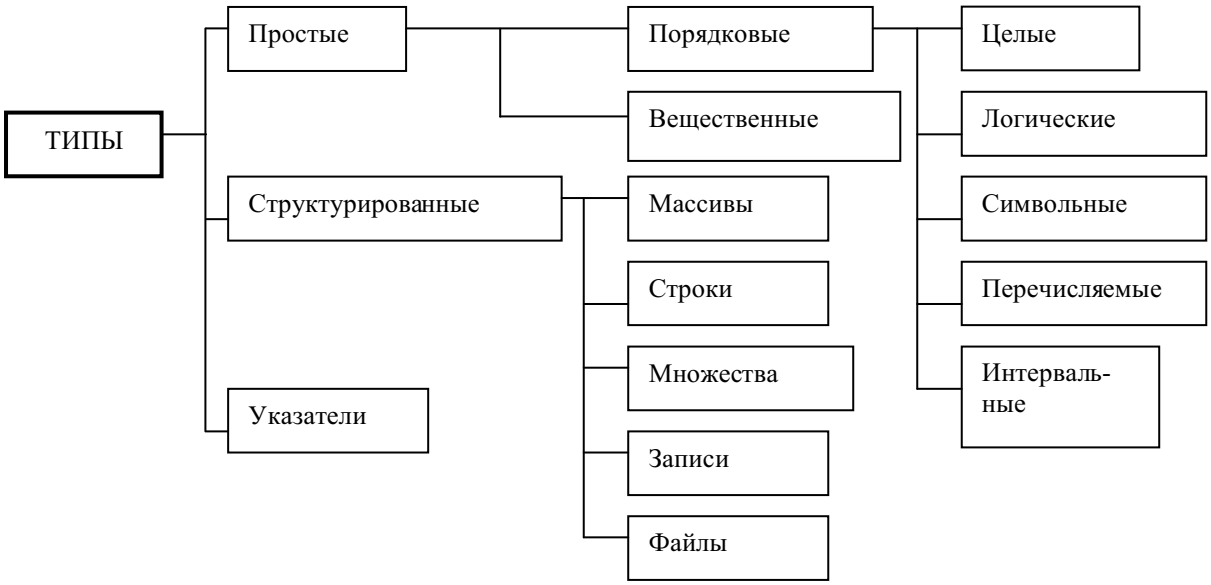

Рисунок 2. Классификация типов данных

Тип данных определяет свойства описываемой величины: *форму ее внутреннего представления, множество принимаемых значений, множество допустимых операций.*

Рассмотрим более подробно простые типы данных:

Таблица 2. Простые типы данных

Идентификатор	Длина, байт	Диапазон (множество) значений
Целые типы		
Integer	2	-32768…32767
Byte	1	0…255
Word	2	0…65535
Shortint	1	-128…127
Longint	4	-2147483648…2147483647
Вещественные типы		
Real	6	$2{,}9\cdot10^{-39}$…$1{,}7\cdot10^{38}$ (11-12)
Single	4	$1{,}5\cdot10^{-45}$…$23{,}4\cdot10^{38}$ (7-8)
Double	8	$5\cdot10^{-324}$…$1{,}7\cdot10^{308}$ (15-16)
Extended	10	$3{,}4\cdot10^{-4932}$…$1{,}1\cdot10^{4932}$ (19-20)
Логический тип		
Boolean	1	True, False
Символьный тип		
Char	1	Все символы кода ASCII

5. Операции и функции. Арифметические выражения

К арифметическим типам данных относятся группы вещественных и целых типов. К ним применимы арифметические операции и операции отношений.

Таблица 3. Бинарные арифметические операции

Знак	Выражение	Типы операндов	Тип результата	Операция
+	A+B	*R, R* *I, I* *I, R; R, I*	*R* *I* *R*	Сложение
-	A-B	*R, R* *I, I* *I, R; R, I*	*R* *I* *R*	Вычитание
*	A*B	*R, R* *I, I* *I, R; R, I*	*R* *I* *R*	Умножение
/	A/B	*R, R* *I, I* *I, R; R, I*	*R* *I* *R*	Вещественное деление
Div	A div B	*I, I*	*I*	Целое деление
Mod	A mod B	*R, R*	*I*	Остаток от деления

К арифметическим величинам могут быть применены стандартные функции Паскаля.

Таблица 4. Математические стандартные функции

Обращение	Тип аргумента	Тип результата	Функция
Pi	-	*R*	Число π
Abs(x)	*I, R*	*I, R*	Модуль аргумента x
Arctan(x)	*I, R*	*R*	Арктангенс x (радианы)
Cos(x)	*I, R*	*R*	Косинус x (x в радианах)
Exp(x)	*I, R*	*R*	e^x - экспонента
Frac(x)	*I, R*	*R*	Дробная часть x
Int(x)	*I, R*	*R*	Целая часть x
Ln(x)	*I, R*	*R*	Натуральный логарифм x
Random	-	*R*	Псевдослучайное число в интервале [0,1)
Random(x)	*I*	*I*	Псевдослучайное число в интервале [0,x)
Round(x)	*R*	*I*	Округление до ближайшего целого
Sin(x)	*I, R*	*R*	Синус x (в радианах)
Sqr(x)	*I, R*	*I, R*	Квадрат x
Sqrt(x)	*I, R*	*R*	Корень квадратный из x
Trunc(x)	*R*	*I*	Ближайшее целое, не превышающее x по модулю

Возведение в степень осуществляется следующим образом:

$$a^b = \exp(b*\ln(a))$$

Арифметическое выражение задает порядок выполнения действий над числовыми величинами. Арифметические выражения содержат арифметические операции, функции, операнды, круглые скобки.

К арифметическим операторам относят оператор присваивания (:=), который имеет следующую структуру:

<Переменная> := <Арифметическое выражение>

Пример 3. Создать программу, которая вычисляла бы корень квадратный заданного числа (использовать функцию SQRT).

```
program Z2;
var a:real;
begin
   writeln('введите число');
   read(a);
   writeln('квадрат числа равен',sqrt(a):5:1);
end.
```

Задание 1. Вычислить расстояние между двумя точками с заданными координатами (x1, y1) и (x1, y2).

Задание 2. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем куба.

Задание 3. Написать программу, которая выводит первые четыре степени числа Пи.

6. Логические величины, операции, выражения

Логическое выражение - логическая формула, записанная на языке программирования. Логическое выражение состоит из логических операндов, связанных логическими операциями и круглыми скобками. Результатом вычисления является булева величина, которая, как и переменная логического типа (Boolean), может принимать значения либо true, либо false. При создании логических выражений используются операции отношения (> < >= <= <>) и логические операции (not (не), xor (исключающее или), or (или), and (логическое и)).

Упражнения

Вычислить значения логических выражений:

1. K mod 7 = K div 5 – 1 при K=15;
2. t and (P mod 3 = 0) при t=true, P=10101;
3. (x*y<>0) and (y>x) при x=2, y=1;
4. a or not b при a=false, b=true.

7. Операторы языка

7.1. Составной оператор

Составной оператор - последовательность произвольных операторов программы, заключенная в операторные скобки - зарезервированные слова **Begin…End**. Среди составного оператора могут быть другие операторы, TP допускает произвольную глубину их вложенности:

```
Begin
 …….
   Begin
   …….
       Begin


       End;
      ……..
   End;
 ……
End.
```

7.2 Условный оператор

Позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить то или иное действе; средство ветвления вычислительного процесса.

Общий вид команды ветвления в виде блок-схемы следующий:

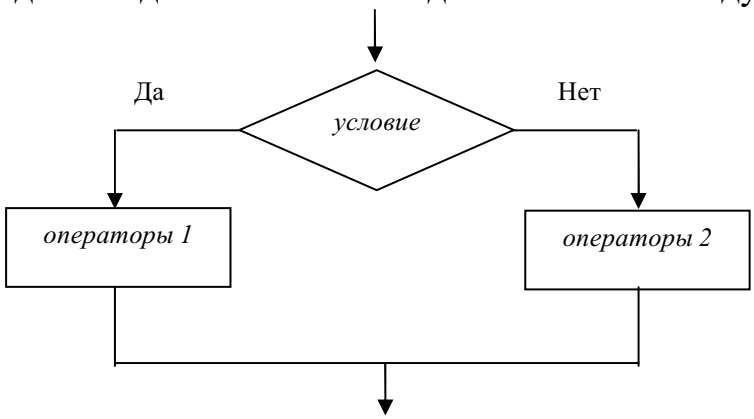

Рисунок 3. Вид команды ветвления в виде блок-схемы

Структура:

IF <условие> **THEN** <оператор1> [**ELSE** <оператор2>]

<условие> - произвольное выражение логического типа;
<оператор1>, <оператор2> - любые операторы языка TP
ELSE - необязательный параметр.

Пример 4. Найти площадь треугольника, используя формулу Герона, учесть следующие ограничения: стороны треугольника - положительные числа, каждая из сторон меньше суммы двух других.

```
var a,b,c,p,s:real;
BEGIN
writeln('Введите длины сторон треугольника');
write('a='); readln(a);
write('b='); readln(b);
write('c='); readln(c);
if (a>0) and (b>0) and (c>0) and (a<=b+c)
                  and (b<=a+c) and (c<=a+b) then
 begin
  p:=(a+b+c)/2;
  s:=sqrt(p*(p-a)*(p-b)*(p-c));
  writeln('s=',s:5:2);
 end
else writeln ('Неверные исходные данные');
END.
```

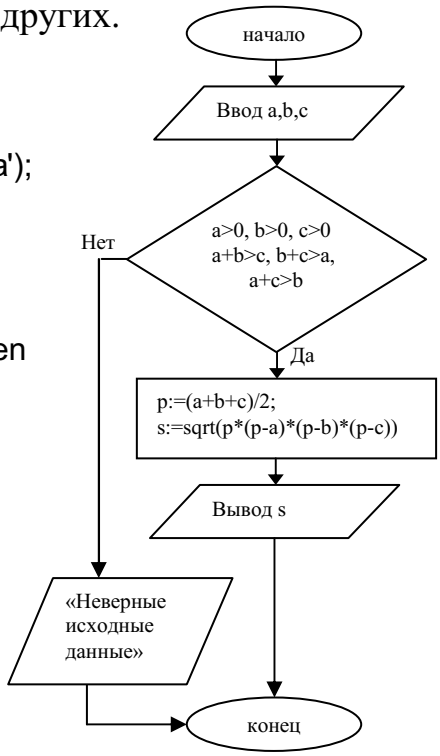

Рисунок 4

Задание 4. Создать программу, которая проверяла бы введенное число на четность (используйте оператор mod).

Типичные задачи, решаемые в курсе информатики по данной теме*:*

1. Задачи на составление логического выражения.
2. Задачи на определение принадлежности точки к указанной закрашенной области.
3. Вычислить значение функции вида: $F(x)=\begin{cases} x^2-3x+9, & x\le 3 \\ \dfrac{1}{x^3+6}, & x>3 \end{cases}$.

Задание 5. Определить правильность даты, введенной с клавиатуры (число – от 1 до 31, месяц – от 1 до 12). Если введены некорректные данные, то сообщить об этом.

Задание 6. Дана точка A(x, y). Определить, принадлежит ли она треугольнику с вершинами в точках (x1, y1), (x2, y2), (x3, y3).

7.3. Оператор выбора

Позволяет выбрать одно из нескольких возможных продолжений программы. Параметром, по которому осуществляется выбор, служит ключ выбора.

На языке Turbo Pascal структура имеет следующий вид:

CASE <ключ_выбора> **OF** <список_выбора> [**ELSE** <операторы>] **END**

<ключ_выбора> - ключ выбора;
<список_выбора> - одна или более конструкций вида:
<константа_выбора> **:** <оператор>;
<константа_выбора> - константа того же типа, что и <ключ_выбора>
<операторы> - произвольные операторы ТР.

Пример 5. Программа при вводе одного из символов y или Y выведет на экран слово ДА, а при вводе n или N – НЕТ.

```
Var
  Ch:char;
Begin
  Readln(ch);
  Case ch of
    ‘n’, ‘N’: writeln(‘НЕТ’);
    ‘y’, ’Y’: writeln(‘ДА’);
  end
End.
```

Задание 7. Составить программу, позволяющую получить словесное описание школьных отметок (1 – «плохо», 2 – «неудовлетворительно», 3 – «удовлетворительно», 4 – «хорошо», 5 – «отлично»).

Задание 8. «Калькулятор». Пользователь вводит с клавиатуры два числа и символ арифметического действия (+, -, *, /), компьютер должен напечатать результат.

7.4. Операторы повторения

Цикл - главное средство заставить компьютер много раз сделать одно и то же или похожее.

7.4.1. Счетный оператор цикла FOR (цикл по параметру)

Вид команды в виде блок-схемы следующий:

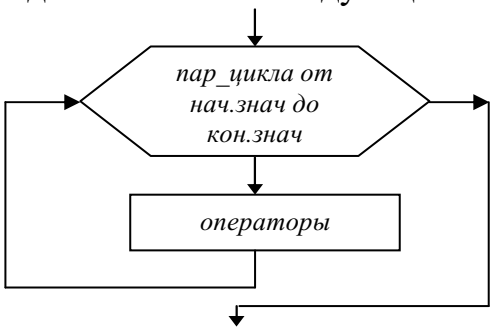

Рисунок 5. Вид команды "цикл со счетчиком" в виде блок-схемы

Общий вид конструкции следующий:

FOR <пар_цикла>**:=**<нач_знач> **TO** <кон_знач> **DO** <оператор>

или

FOR <пар_цикла>**:=**<нач_знач> **DOWNTO** <кон_знач> **DO** <оператор>

Выполнение оператора **FOR** в первом варианте происходит по следующей схеме:

1. Вычисляются значения <нач_знач> и <кон_знач>. Это делается только раз при входе в цикл.
2. Параметру цикла (<пар_цикла>) присваивается значение <нач_знач>.
3. Значение параметра цикла сравнивается со значением <кон_знач>. Если параметр цикла меньше или равен этому значению, то выполняется тело цикла, в противном случае выполнение цикла заканчивается.
4. Значение параметра изменяется на следующее значение в его типе (для целых – увеличивается на единицу); происходит возврат к пункту 3.

Слово **DOWNTO** буквально можно перевести как «вниз до». В таком случае параметр цикла изменяется по убыванию.

Пример 6. Составить программу вычисления факториал целого положительного числа.

```
Program factorial_for;
var i:word;
n:integer;
factorial:longint;
BEGIN
  writeln('введите число N'); readln(n);
  factorial:=1;
  for i:=1 to n do factorial:=factorial*i;
  writeln ('факториал числа N = ',factorial);
END.
```

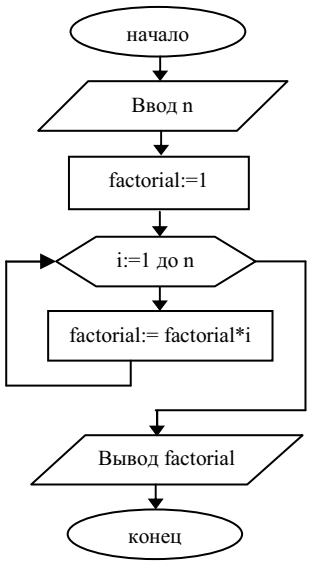

Рисунок 6

Задание 9. Начав тренировки, спортсмен в первый день пробежал 10 км. Каждый день он увеличивал дневную норму на 10% нормы предыдущего дня. Какой суммарный путь пробежит спортсмен за 7 дней?

Задание 10. Составить программу, которая печатает таблицу умножения натуральных чисел.

7.4.2. Оператор цикла WHILE (с предпроверкой условия)

Выполнение серии команд (тела цикла) повторяется, пока условие цикла истинно. Когда условие становится ложным, цикл заканчивает выполнение.

Общий вид команды в блок-схеме следующий:

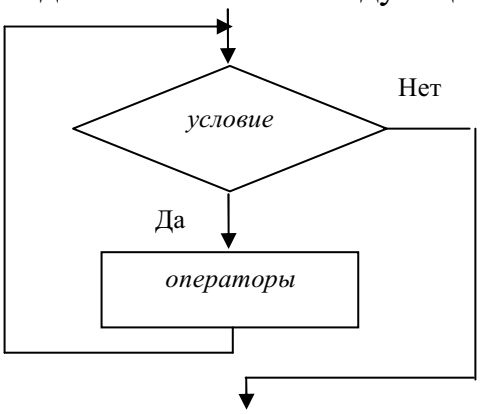

Рисунок 7. Вид команды цикла с предпроверкой условия в виде блок-схемы

На языке Turbo Pascal структура имеет следующий вид:

WHILE <условие> **DO** <оператор>

<условие> - произвольное выражение логического типа;
<оператор> - произвольный оператор языка ТР.

Пример 7. Задачу примера 6 решить с помощью оператора While…Do

```
program p1;
var n,i:integer;
      f:longint;
begin
  write('n='); readln(n);
  i:=0; f:=1;
  while i<n do
  begin
      i:=i+1;
      f:=f*i;
  end;
  writeln('factorial n ',f);
end.
```

7.4.3. Оператор цикла REPEAT…UNTIL (с постпроверкой условия)

В цикле с постпроверкой условия используется условие окончание цикла. Когда оно становится истинным, цикл заканчивает работу.

Общий вид команды в блок-схеме следующий:

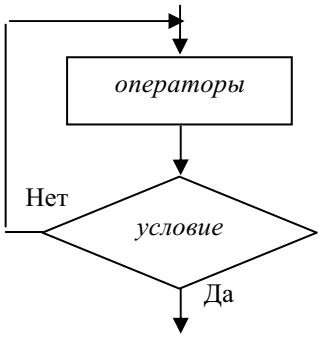

Рисунок 8. Вид команды цикла с постпроверкой условия в виде блок-схемы

На языке Turbo Pascal структура имеет следующий вид:

REPEAT <оператор> **UNTIL** <условие>

<оператор> - произвольный оператор языка ТР;
<условие> - произвольное выражение логического типа.

Задание 11. Задачу примера 6 решить с помощью оператора Repeat…Until.

Пример 8. Составить программу, которая запрашивает пароль (например, четырехзначное число), до тех пор, пока пароль не введен правильно, зачем появляется текст «Пароль введен верно».

```
var p:string;
const parol='1234';
begin
repeat
      begin
        writeln('введите пароль'); read(p);
        readln;
      end
until p=parol;
writeln('Пароль введен верно');
readln;
end.
```

Задание 12. Дано натуральное число N. Найти сумму его цифр.

Задание 13. Найти наибольшую и наименьшую цифры в записи данного натурального числа.

Задание 14. Составить программу перевода числа из двоичной системы счисления в десятичную.

8. Порядковый тип данных

К порядковым типам относятся *целые*, *логический*, *символьный*, *перечисляемый* и *тип-диапазон*.

К любому из них применимы функции:
Ord (a) – возвращает порядковый номер значения выражения;
Pred(a) – предыдущее значение порядкового типа;
Succ(a) – следующее значение порядкового типа.

8.1. Символьный тип данных

Значением символьного типа является множество всех символов ПК. Каждому слову приписывается целое число в диапазоне 0...255. Это число служит кодом внутреннего представления символа, его возвращает функция Ord. Для кодировки используется код ASCII (American Standart Code for Information Interchange – американский стандартный код для обмена информацией). Это 7-битный код. В то же время в 8-битном байте, отведенном для хранения символов в TP, можно закодировать в 2 раза больше символов в диапазон от 0 до 255. Первая половина символов ПК соответствует стандарту ASCII. Вторая не ограничена жесткими рамками стандарта и может применяться на ПК разных типов. Символы с кодами 0…31 относятся к служебным кодам. Если эти коды используются в символьном тексте, то они считаются пробелами. К типу Char применимы следующие операции отношения, а также встроенные функции:
Chr(b) – функция типы Char, преобразует выражение b типа Byte в символ и возвращает его своим значением;
Upcase (ch) – возвращает прописную букву.

Пример 9. Введенную строчную букву преобразовать в прописную.

```
program pr;
var a:char;
begin
 write('введите строчную английскую букву ');
 readln(a);
 writeln(upcase(a));
 readln;
end.
```

8.2. Перечисляемый тип данных

Задается перечислением тех значений, которые он может получать.

Объявление: Var Manth: (Jen, Feb, March, April, May, June, July, August, Sep, Oct, Nov, Dec). Переменная Manth может принимать только одно из перечисленных в скобках значений. Эти значения не являются строками, их нельзя вывести на печать, однако их удобно применять при программировании.

8.3. Логический тип данных

Переменная логического типа может принимать только два значения: false и true.

8.4. Тип – диапазон

Тип-диапазон есть подмножество своего базового типа, в качестве которого может выступать любой порядковый тип. Задается границами своих значений внутри базового типа.

Пример 10. С клавиатуры вводятся две даты. Выдать сообщение «Правильно», если первая введенная дата предшествует второй «Неправильно», если наоборот.

```
program pr;
uses crt;
var
 dn1,dn2:1..31;
 mes1,mes2:1..12;
 god1,god2:1900..2100;
begin
 write('введите первую дату');
 readln(dn1,mes1,god1);
 write('введите вторую дату');
 readln(dn2,mes2,god2);
 if (((dn1<dn2) or (mes1<mes2)) and (god1=god2)) or (god1<god2) then_
writeln('Правильно') else writeln('Неправильно');
 repeat until keypressed;
end.
```

Задание 15. Изменить предыдущую задачу следующим образом: даты рождения двух человек, по введенным данным определить кто старше.

9. Подпрограммы

Очень часто в процессе выполнения программы требуется многократное выполнение какой-либо ее части. Логичнее всего эту часть программы выделить, обозначить ее имя, задать ей необходимые параметры и вызвать эту часть по ее имени. Для этого используются процедуры и функции. Все процедуры и функции подразделяются на две группы: встроенные (стандартные) и пользовательские. Стандартные подпрограммы входят в состав языка и вызываются для выполнения по строго фиксированному имени. Использовать имена встроенных процедур и функций для именования собственных нельзя. Встроенные процедуры и функции расположены в специальных библиотечных модулях, которые имеют системные имена (system - основной модуль, crt - средства управления дисплеем и клавиатурой, dos - средства реализации различных функций DOS, graph - пакет графических средств…). В свою очередь, подключаемые модули подключаются с помощью зарезервированного слова Uses.

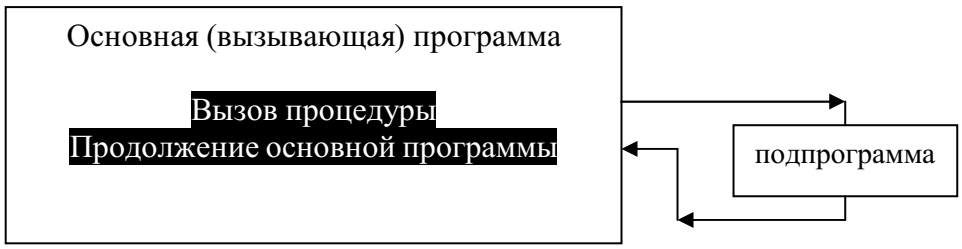

Рисунок 9. Взаимодействие вызывающей программы и подпрограммы

9.1. Процедуры пользователя

Процедура - независимая поименованная часть программы, которую можно вызвать по ее имени для выполнения определенных действий. Структура процедуры такая же, как и структура программы.

Описание процедуры включает заголовок (имя) и тело процедуры. Заголовок процедуры состоит из зарезервированного слова Procedure, имени процедуры и списка *формальных параметров*. Список формальных параметров необязателен и может отсутствовать, если же он есть, то в нем должны быть перечислены имена формальных параметров и их типы, например: Procedure sb (a: real; b: integer; c:char);

Имя процедуры создается по тем же правилам, что и имена переменных. Все, что идет после имени процедуры, называют телом процедуры. Синтаксис описания процедуры имеет следующий вид:

```
Procedure <имя> [(список формальных параметров)];
  Const…;
  Var…;
  Begin
    <Операторы>;
  End;
```

Обмен аргументами и результатами между основной программой и процедурой производится через *формальные* и *фактические параметры*, существует еще один механизм обмена - через глобальные переменные. Если подпрограмма описана с *формальным параметрами*, то обращение к ней производится оператором подпрограммы с *фактическими параметрами*. Правила соответствия между формальными параметрами и фактическими: соответствие по количеству, по последовательности, по типам.

Пример 11. Вывести на экран все четные числа до указанного. Выполните программу в режиме пошагового выполнения, используйте функциональную клавишу F7.

```
var n,i:word;
procedure pr(x:word);
begin
```

```
 if x mod 2=0 then write('  ',x);
end;
BEGIN
 writeln('введите число'); read(n);
 for i:=1 to n do pr(i);
 readln;
 readln;
END.
```

9.2. Пользовательские функции

Ранее мы сталкивались со стандартными функциями: sqr(x). Возможно создавать собственные функции. Функция очень похожа на процедуру. В некоторых языках нет понятия процедуры. Ее заменяет функция как более общая подпрограмма. Функция отличается от процедуры тем, что результат ее работы возвращается в виде значения этой функции, и, следовательно, вызов функции может использоваться наряду с операндами и выражениями.

```
Function <имя> [(список формальных параметров)]: <тип>;
   Const…;
   Var…;
   Begin
     <Операторы>;
   End;
```

Пример 12. Составить программу вычисления площади кольца по значениям внутреннего и внешнего радиусов, используя подпрограмму вычисления площади круга.

```
var r1,r2:integer;
    function p(r:integer):real;
    var p1:real;
    const pi=3.14;
    begin
     p1:=pi*sqr(r);
     p:=p1;
    end;
BEGIN
   write('r1-? '); readln(r1);
   write('r2-? '); readln(r2);
   writeln(abs(p(r1)-p(r2)));
END.
```

9.3. Локальные и глобальные переменные.

Деление переменных на глобальные и локальные является способом повысить надежность больших программ и понизить вероятность запутаться при их написании. Областью действия описания любого

программного объекта (переменной, типа, константы …) является тот блок, в котором расположено это описание. Если данный блок вложен другой (подпрограмма), то присутствующие в нем описания являются *локальными*. Они действую только в пределах внутреннего блока. Описания же, стоящие во внешнем блоке, называют *глобальными*.

Языки, в которых предусмотрены механизмы структурирования программ посредством подпрограмм, называются ПРОЦЕДУРНО-ОРИЕНТИРОВАННЫМИ.

Пример 13. По координатам вершин треугольника вычислить его периметр, используя подпрограмму вычисления длины отрезка, соединяющего две точки.

```
program p15;
var x1,x2,x3,y1,y2,y3:integer;
    p,s,a,b,c:real;
  function dlina(a1,b1,a2,b2:integer):real;
  var p1:real;
  begin
   p1:=sqrt(sqr(a1-a2)+sqr(b1-b2));
   dlina:=p1;
  end;
BEGIN
 write('x1,y1-?'); readln(x1,y1);
 write('x2,y2-?'); readln(x2,y2);
 write('x3,y3-?'); readln(x3,y3);
 a:=dlina(x1,y1,x2,y2);
 b:=dlina(x1,y1,x3,y3);
 c:=dlina(x3,y3,x2,y2);
 p:=a+b+c;
 writeln ('p= ',p);
END.
```

Задание 16. Составить программу для вычисления суммы факториалов всех четных чисел от *m* до *n*.

Задание 17. Даны три целых числа. Определить, сумма цифр которого из них больше. Подсчет суммы цифр организовать через подпрограмму.

10. Структурированные типы данных

Неструктурированные типы данных имеют следующее отличие от структурированных: одно имя – одно значение.

Классифицируются структурированные типы данных по следующим признакам:

- Однородность (элементы однотипны);
- Упорядоченность (между элементами определен порядок следования);

- Тип доступа (прямой или последовательный);
- Статическая или динамическая.

10.1. Массивы

Массив - однородная упорядоченная статическая структура прямого доступа. Массив можно назвать совокупностью фиксированного числа одинаковых компонент, каждая из которых снабжается индексом. Для того чтобы описать массив необходимо задать тип компонента и тип индекса.

Var <идентификатор>: **Array** [<тип индекса>] **of** <тип компонент>;

Чаще всего, в качестве типа индекса употребляется интервальный тип, может быть и перечисляемый тип, а также любым скалярным порядковым типом.

Пример объявления массива:

```
Var mass=array[1..10] of real; {пример описания одномерного}
                               {массива}
Var a[1..15,1..4] of integer;  {пример описания двумерного}
                               {массива}
```

Все задачи можно условно представить из нескольких основных блоков:

- Блок ввода значений элементов массива (формирование элементов массива случайным образом).
- Блок вывода исходного массива.
- Блок обработки массива.
- Блок вывода результатов обработки.

Наиболее часто используемые массивы при решении задач: одномерные, двумерные.

Пример 14. Задать одномерный целочисленный массив [1..5], а затем вывести его на экран.

```
var a:array [1..5] of integer;     {объявление массива}
i:integer;                         {объявление переменной i}
                                   {кот. определяет индекс элемента массива)}

BEGIN
 for i:=1 to 5 do                  {формирование массива}
  begin
  write('?',i,' ');
  readln(a[i]);
  end;
 for i:=1 to 5 do write(a[i],' '); {вывод элементов массива на экран}
 readln;                           {задержка экрана}
END.
```

Пример 15. Составить программу, в которой элементы массива [1..100] формируются случайным образом. Вывести массив на экран.

```
const n=100;
var a:array [1..n] of integer;
   i:integer;
BEGIN
randomize;                    {настройка генератора случайных чисел}
 for i:=1 to n do             {формирование массива и вывод на экран}
   begin
        a[i]:=random(100);    {генерация случайного числа от 0 до 100}
        write(a[i],' ');
   end;
readln;
END.
```

Пример 16. Составить программу, в которой элементы двумерного массива [1..n, 1..m] формируются случайным образом. Вывести массив на экран.

```
const n=3;
       m=4;
var a:array [1..n,1..m] of integer;
   i,j:integer;
BEGIN
randomize;
 for i:=1 to n do
    begin
      for j:=1 to m do
          begin
           a[i,j]:=-100+random(200);   {генерация случайного числа от -100 до 100}
           write(a[i,j],' ');
          end;
    writeln;
    end;
readln;
END.
```

Пример 17. *Поиск элементов массива*. Найти наибольший элемент двумерного массива вещественных чисел.

```
const n=3;
     m=4;
var a:array [1..n,1..m] of integer;
   i,j,max:integer;
BEGIN
randomize;
 max:=-100;
 for i:=1 to n do
 begin
  for j:=1 to m do
  begin
  a[i,j]:=-100+random(200);
  write(a[i,j],' ');
  if a[i,j]>max then max:=a[i,j];
  end;
 writeln;
 end;
writeln('max = ',max);
readln;
END.
```

Задание 18. *Поиск элементов массива*. Найти наименьший элемент двумерного массива вещественных чисел.

Пример 18. *Проведение арифметических операций над элементами массива.* Найти сумму положительных элементов двумерного массива целых чисел.

```
const n=3;
      m=3;
var a:array [1..n,1..m] of integer;
   i,j,summ:integer;
BEGIN
randomize;
 for i:=1 to n do
 begin
  for j:=1 to m do
  begin
  a[i,j]:=-10+random(20);
  write(a[i,j],' ');
  if a[i,j]>0 then summ:=summ+a[i,j];
  end;
 writeln;
 end;
writeln('summ = ',summ);
readln;
END.
```

Задание 19. *Замена элементов массива.* В одномерном массиве целых чисел заменить все нули единицами.

Задание 20. Составить программу, в которой сформировать двумерный целочисленный массив случайных чисел A[1..50,1..50] и подсчитать количество отрицательных и положительных чисел в нем.

Задание 21. Составить программу, в которой ввести двумерный целочисленный массив A[1..4,1..4] следующим образом: A[1,1], A[1,2], A[1,3], A[1,4], A[2,1],…, A[3,4], A[4,4]. Распечатать его: A[1,1], A[2,1], A[3,1], A[4,1], A[1,2], A[2,2],…, A[4,3], A[4,4].

10.2. Строки

Строка - однородная упорядоченная статическая структура прямого доступа. Строка - последовательность символов. Каждый символ занимает 1 байт памяти (код ASCII). Количество символов в строке называется ее длиной. Длина строки может находиться в диапазоне от 0 до 255. Строковые величины могут быть константами и переменными. *Строковая константа* - последовательность символов, заключенная в апострофы. *Строковая переменная* описывается в разделе описания переменных:

VAR <идентификатор>: **string** [<максимальная длина строки>];

по умолчанию длина строки - 255.

Строковая переменная занимает в памяти компьютера на 1 байт больше, чем длина строки. Нулевой байт содержит значение *текущей длины строки*. По мере заполнения строки ее текущая длина возрастает.

Символы внутри строки индексируются (нумеруются) от единицы. Каждый элемент идентифицируется именем строки с индексом: st[2]. Над

строковыми данными допустимы операции сцепления и операции отношения: +, =, <, >, <>. Основные строковые функции и процедуры:
Функция *Length(s)* - определяет текущую длину строки. Результат - значение целого типа.
Функция *Copy(s,poz,n)* - выделяет из строки s подстроку длиной в n символов, начиная с позиции poz.
Функция *Concat(s1,s2,…sn)* - выполняет сцепление строк в одну.
Функция *Pos(s1,s2)* - обнаруживает первое появление в строке s2 подстроки s1. Результат - целое число.
Процедура *Delete(s,poz,n)* - выполняет удаление n символов из строки s, начиная с позиции poz.
Процедура *Insert(s1,s2,poz)* - выполняет вставку строки s1 в строку s2, начиная с позиции poz.
Процедура *Str(x,s)* - заданное числовое значение преобразует в строку символов.
Процедура *Val(s,x,c)* - строка символов s, состоящая из цифр преобразуется в число. Значение передается переменной x. Параметр *c* содержит ноль, если преобразование прошло успешно, в противном случае он содержит номер позиции в строке, где обнаружена ошибка.

Пример 19. Подсчитать количество пробелов в строке.

```
var s:string;
  kol,i:integer;
BEGIN
 write('? ');readln(s);
 kol:=0;
 for i:=1 to length(s) do
  begin
   if s[i]=' ' then kol:=kol+1;
  end;
 writeln('kol = ',kol);
END.
```

Пример 20. В веденном слове заменить первую и последнюю буквы.

```
var s:string;
  i:integer;
  a:char;
BEGIN
 write('? ');readln(s);
 a:=s[1];
 s[1]:=s[length(s)];
 s[length(s)]:=a;
 writeln(s);
 readln;
END.
```

Пример 21. Перевернуть введенное слово.

```
var s:string;
  i:integer;
BEGIN
 write('? ');readln(s);
 for i:=length(s) downto 1 do
  begin
   write(s[i]);
  end;
 readln;
END.
```

Задание 22. Дана строка. Определить, сколько раз входит в нее группа букв abc.

Задание 23. Дано слово. Вставить пробел после каждого символа.

Некоторые процедуры и функции модуля CRT

Процедура *ClrScr* - очищает экран.
Функция KeyPressed - возвращает значение типа boolean, указывающее состояние буфера клавиатуры: False - пуст, True - в буфере есть хоть один символ.
Функция ReadKey - возвращает значение типа char. Код символа из буфера клавиатуры.

Пример 22. Программа определяет расширенный код любой клавиши. Для завершения работы программы нужно нажать клавишу esc.

```
uses crt;
var c:char;
begin
clrscr;
 repeat
  c:=readkey;
  if c<>#0 then
   writeln(ord(c))
  else
   writeln('0',ord(readkey):8
 until c=#27
end.
```

10.3. Множества

Одним из фундаментальных разделов математики является теория множеств. Некоторые моменты математического аппарата этой теории реализованы в Паскале через множественный тип данных (множества). Множеством называется совокупность однотипных элементов, рассматриваемых как единое целое, однородная не упорядоченная статическая структура. В Паскале могут быть только конечные множества, могут содержать от 0 до 255 элементов. В отличие от массива элементы множества не пронумерованы, не упорядочены. Каждый элемент множества не идентифицируется, и с ним нельзя выполнить никакое действие. Действия могут выполняться только над множеством в целом. Тип элементов множества называют базовым типом (скалярный, за исключением типа real).
[3,4,7,9,12] - множество пяти целых чисел;
[1..100], ['a','b','c'];
[] - пустое множество;
Множества [1,2,3] и [3,1,2]; [1,2,3,4,5] и [1..5] - равнозначны.

Переменные множественного типа *описываются* так:

Var <идентификатор>: **set of** <базовый тип>;

Пример описания множеств:

```
Var A,D: set of Byte;
    B: set of 'a'..'z';
    C: set of Boolean;
```

Нельзя вводить значения во множественную переменную оператором ввода и выводить оператором вывода, множественная

переменная может получить конкретное значение только в результате выполнения оператора присваивания:

<множественная переменная> **:=** <множественное выражение>

```
A:=[50,100,150,200];
B:=['m','n','k'];
C:=[true,false];
```

Операции над множествами

Объединение - множество, состоящее из всех элементов, принадлежащих, одновременно объединяемым множествам (+). [1,2,3,4]+[3,4,5,6]→[1,2,3,4,5,6]

Пересечение множеств (*) - Пересечением двух множеств A и B называется множество, состоящее из элементов, принадлежащих одновременно множеству A и B.

Разность множеств (A-B) - множество, состоящее из элементов множества A, не принадлежащих множеству B.

Операции отношения: =, <>, <=, >=;

Операция вхождения IN - устанавливает связь между множеством и скалярной величиной, тип которой совпадает с базовыми типом множества. Если x - такая скалярная величина, а M - множество, то операция вхождения записывается так: **x In** M. Результат - логическая величина, true - если входит в множество, false - в противном случае.

Процедура *INCLUDE(s,i)* – включает новый элемент *i* в множество *s*.

Процедура *EXCLUDE(s,i)* – исключает элемент *i* из множества *s*.

Пример 23. Дана символьная строка. Подсчитать в ней количество знаков препинания (. , ! ; : - ?).

```
uses crt;                          {подключение модуля CRT}
var s:string;
    i,kol:integer;
    a: set of char;                {описание множества a}
BEGIN
write('? '); readln(s);            {запрос и ввод строки}
a:=[',','.',':',';','!','?','-'];  {формирование множества a – множество}
                                   {знаков препинания}
kol:=0;
for i:=1 to length(s) do           {счет знаков препинания в строке}
 begin
   if s[i] in a then kol:=kol+1;
 end;
writeln('kol = ',kol);
repeat until keypressed;           {задержка экрана}
END.
```

Пример 24. Задано множество целых положительных чисел от 1 до n. Создать из элементов этого множества такие подмножества, элементы которых удовлетворяют следующим условиям:

a. Элементы подмножества не больше 10;
b. Элементы подмножества кратны 8;
c. Элементы подмножества не кратны 6.

```
uses crt;
const n=100;
var i:integer;
    a,b:set of byte;              {описание множеств a и b}
                                  {a – множество целых чисел от 1 до n}
                                  {b – подмножество множества a}
BEGIN
for i:=0 to n do include(a,i);    {формирование множества a}
for i:=0 to n do                  {формирование множества b}
 begin
   if (i<10) and (i in a) then include(b,i);
 end;
for i:=0 to n do                  {вывод на экран множества a}
 begin
  if i in a then write(i,' ');
 end;
writeln;
for i:=0 to n do                  {вывод на экран множества b}
 begin
  if i in b then write(i,' ');
 end;
repeat until keypressed;          {задержка экрана}
END.
```

Задание 24. Составить программу подсчета количества различных цифр в десятичной записи натурального числа.

Задание 25. Напечатать в убывающем порядке все цифры, не входящие в запись данного натурального числа.

10.4. Описание типов

Описание переменных относящихся к структурированному типу данных, изученных ранее, может быть выполнено двумя способами:

Var <имя переменной>**:** <тип данных>

Type <имя типа>…
Var <имя переменной>**:** <тип данных>;

Примеры:

var a:array [1..n] of integer;	type mass = array [1..n] of integer; var a:mass;
var s:string[10];	type stroka = string[10]; var s: stroka;
var A,D: set of Byte;	type g = set of Byte; var A,F:g;

Задание 26. Измените описания переменных в примерах 14, 17, 24 предварительно описав типы используемых переменных.

10.5. Записи

Запись – неоднородная упорядоченная статическая структура прямого доступа. Запись - это тип данных, состоящий из конечного числа компонент или полей. Поля могут быть различного типа.
Пример: анкетные сведения о студенте

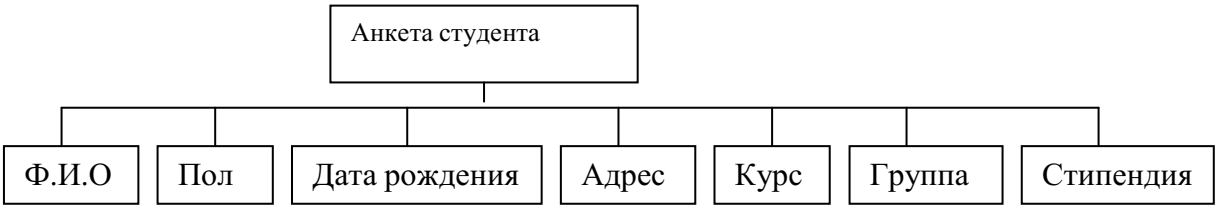

Рисунок 10.

Такая структура называется двухуровневым деревом.

Общий вид объявления такого типа данных:

```
Type
<Название_типа> = record
<Имя_поля1>: <тип поля1>;
<Имя_поля2>: <тип поля2>; …
end;
```

Описание записи «анкетные данные о студенте» в программе будут выглядеть следующим образом:

```
Type Anketa1 = record
     FIO: String[50];
     Pol: Char;
     Dat: String[16];
     Adres: String[50];
     Curs: 1..5;
     Grup:1..10;
     Stip: Real;
end;
Var Student: Anketa1;
```

К каждому элементу записи можно обратиться, используя составное имя, которое имеет следующую структуру;

<имя переменной>.<имя поля>

Student.FIO
Student.Dat

Если требуется полю курс присвоить значение 3, то это делается так:

Student.curs:=3;

Поля записи могут иметь любой тип, в частности, сами могут быть записями. Тогда реализуется многоуровневые деревья. В программе могут быть использовать массивы записей.

Пример 25. В столовой предлагается N комплексных обедов, состоящих из четырех блюд. Известна стоимость каждого блюда. Сколько стоит самый дешевый и самый дорогой обед?

```
uses crt;
type obed=record
   pervoe:integer;
   vtoroe:integer;
   tretie:integer;
   desert:integer;
end;
const n=3;
var a:array[1..n] of obed;
    i,max,min,indmax,indmin,sena:integer;
BEGIN
clrscr;
a[1].pervoe:=30;
a[1].vtoroe:=100;
a[1].tretie:=40;
a[1].desert:=50;
a[2].pervoe:=40;
a[2].vtoroe:=110;
a[2].tretie:=20;
a[2].desert:=70;
a[3].pervoe:=1000;
a[3].vtoroe:=80;
a[3].tretie:=30;
a[3].desert:=80;
max:=0;
min:=1000;
for i:=1 to n do
 begin
  sena:=a[i].pervoe+a[i].vtoroe+a[i].tretie+a[i].desert;
  if sena<min then
   begin
    min:=sena;
```

```
    indmin:=i;
   end;
  if sena>max then
   begin
    max:=sena;
    indmax:=i;
   end;
 end;
writeln(indmax,'-й обед самый дорогой, его цена = ',max);
writeln(indmin,'-й обед самый дешевый, его цена = ',min);
repeat until keypressed;
END.
```

Задание 27. Определить, какая из двух точек находится ближе к началу координат (используйте запись POINT с полями X и Y).

10.6. Файлы. Файловые переменные

Понятие "файл", прежде всего, связывают с информацией на устройствах внешней памяти. В Паскале понятие файла употребляется в двух смыслах:

- Как поименованная информация на внешнем устройстве (внешний файл);
- Как переменная файлового типа в Паскаль - программе (внутренний файл).

С элементами файла можно выполнять только две операции: читать из файла и записывать в файл.

Файловый тип переменной - структурированный тип, представляющий собой совокупность однотипных элементов, количество которых заранее (до исполнения программы) не определено (однородная упорядоченная динамическая структура последовательного доступа). Описание файловой переменной выглядит следующим образом:

Var <имя переменной>: **File of** <тип элементов>;

Var <имя переменной> = Text;

Var <имя переменной> = **File**

В зависимости от способа объявления можно выделить три типа файлов:

- типизированные;
- текстовые;
- нетипизированные.

Примеры описания файловых переменных:

```
Var:  Fi: file of integer;
      Fr: file of real;
```

```
Fc: file of char;
Ft = Text;
Fn = File;
```

Файл можно представить как последовательную цепочку элементов, пронумерованных от 0, заканчивающуюся специальным кодом, называемым *маркером конца* (<м.к.>).

Эл. 0	Эл. 1	Эл. 2	…	Эл. N	М.к.

Количество элементов, хранящихся в данный момент в файле – *текущая длина* файла. Существует специальная ячейка памяти, которая хранит адрес элемента файла, предназначенного для текущей обработки (записи или чтения). Этот адрес называется *указателем* или *окном файла*.

Для того чтобы записать в файл, его следует открыть для записи процедурой *Rewrite(FV)*, где *FV* - имя файловой переменной. При этом указатель устанавливается на начало файла. Если в файле есть информация, то она исчезает. Схема выполнения процедуры:

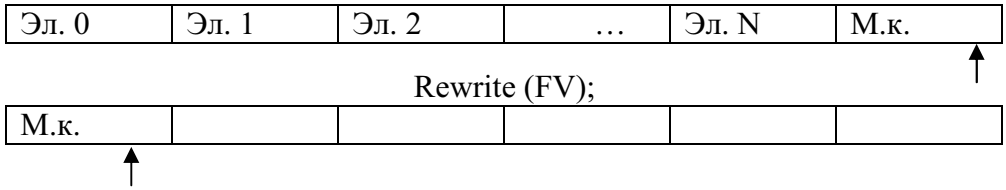

Рисунок 11. Схема выполнения процедуры Rewrite

Запись в файл осуществляется процедурой *Write(FV,V)*, где *V* - переменная того же типа, что и файл *FV*. Запись происходит туда, где установлен указатель. Сначала записывается значение, затем указатель смещается в следующую позицию. Если новый элемент вносится в конец файла, то сдвигается маркер конца.

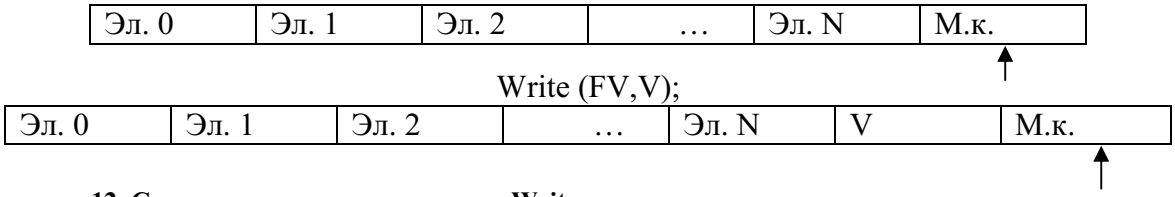

Рисунок 12. Схема выполнения процедуры Write

Для чтения элементов файла его следует открыть для чтения процедурой *Reset (FV)*. В результате указатель устанавливается на начало файла, при этом вся информация в файле сохраняется.

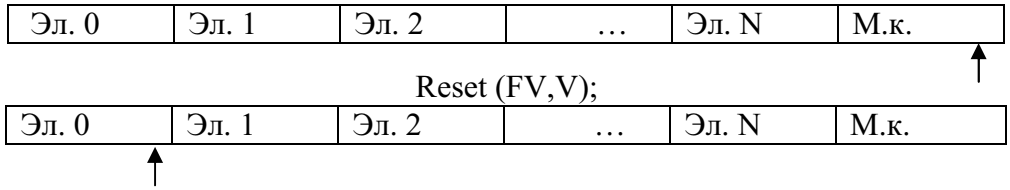

Рисунок 13. Схема выполнения процедуры Reset

Чтение из файла осуществляется процедурой *Read (FV,V)*. Значение текущего элемента файла записывается в переменную *V*; указатель смещается к следующему элементу.

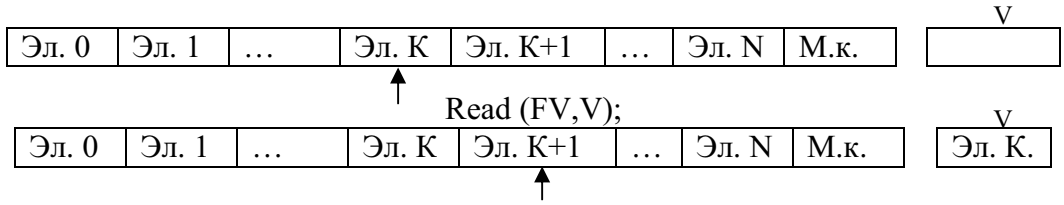

Рисунок 14. Схема выполнения процедуры Read

Для проверки конца файла используется логическая функция *EOF(FV)* (end of file).

Для организации связи между файловой переменной и внешним файлом используется процедура назначения: *Assign(FV, IVF)*, где *IVF* - идентификатор внешнего файла. Например: Assign (Fi,'number.dat');

Работа с файлом в программе завершается его закрытием с помощью процедуры *Close(FV)*.

10.6.1. Типизированные файлы

Пример 26. Создать файл, содержащий среднесуточные температуры. Признаком конца ввода будет число 999. Проверить содержимое созданного файла.

```
var Ft:file of real;
      t: real;
begin
 assign(Ft,'c:\temp.dat');
 rewrite(Ft);{открыть файл для записи}
 writeln('вводите данные');
 writeln('признак конца число 999');
 readln(t);
 while t<>999 do
  begin
    write(Ft,t);
    write('?');
    readln(t);
  end;
 writeln('ввод закончен');
 close(Ft);
reset(Ft); {открыть файл для записи}
  while not eof(Ft) do
  begin
   read(Ft,t);
   write(t:3:5,' ');
  end;
  close(Ft);
  readln;
 end.
```

Пример 27. Определить среднюю температуру для значений, хранящихся в файле temp.dat.

```
var Ft:file of real;
   t,sum: real;
   i:byte;
begin
 assign(Ft,'temp.dat');
 sum:=0;
 reset(Ft);
 while not eof(Ft) do
 begin
 read(Ft,t);
 sum:=sum+t;
 i:=i+1;
end;
close(Ft);
writeln('st = ',(sum/i):3:1);
readln;
end.
```

Задание 28. Дан файл вещественных чисел. Определить количество нулевых значений в этом файле.

Задание 29. Записать в файл последовательного доступа N действительных чисел. Вычислить произведение компонентов файла.

10.6.2. Текстовые файлы

Текстовый файл представляет символьную последовательность, разделенную на строки. файл заканчивается маркером конца файла. Каждая строка заканчивается специальным кодом - *маркером конца строки* (<м.к.с.>). В программах для работы с текстовым файлом наряду с процедурами Read и Write применяются процедуры Readln, Writeln.
Readln(FV,<список ввода>); - процедура читает строку из файла с именем *FV*, помещая прочитанное в переменные из списка ввода.
Writeln(FV,<список вывода>) – записывает в файл *FV* значения из списка вывода, после чего выставляется маркер конца строки.
Для обнаружения конца строки в текстовом файле используется логическая функция *Eoln (FV)*.

Пример 28. Создать текстовый файл Note.txt. Подсчитать количество строк в файле.

```
var note:text;
   k:integer;
begin
 k:=0;
 assign(note,'c:\note.txt');
 reset(note);
 while not eof(note) do
 begin
  readln(note); k:=k+1;
 end;
 writeln('количество строк=',k);
 close(note);
end.
```

Задание 30. Дан текстовый файл. Определить длину самой большой строки.

11. Графические возможности Turbo Pascal

Существует два режима работы с монитором: текстовый (экран разбивается на 25 строк и 80 столбцов, в каждой клетке умещается ровно 1 символ) и графический (экран разбивается на множество точек - пикселей); графический режим требует от компьютера больших ресурсов. В составе ТП есть специальный графический модуль graph, который содержит графические процедуры и функции.

Для переключения в графический режим (инициализации графического режима) служит стандартная процедура

Initgraph(<тип_видеоадаптера>, <номер_граф_режима>, <путь>)

```
var gd,gm: integer;
…
gd:=detect; initgraph(gd,gm, 'c:\tp\bgi');
```

Необходимо всегда в конце программы закрывать графический ражим процедурой **CloseGraph;**

Графический режим монитора считает, что экран разделен на 640 пикселей в ширину, 480 – в высоту.

11.1. Основные процедуры:

Рисование графических примитивов:

Putpixel (x,y:integer;c:word) - рисует точку с координатами x,y, цветом c;
Line (xl,yl,x2,y2:integer) - рисует линию от точки с координатами x1,y1, до точки с координатами x2,y2;
Lineto (x,y:integer);
Rectangle (xl,yl,x2,y2:integer) - рисует рамку, с диагональю x1,y1 -x2,y2;
Bar (x1,y1,x2,y2:integer**)** - рисует прямоугольник с диагональю x1,y1 - x2,y2, закрашенный в соответствии с образцом, установленным процедурой Setfillstyle;
Bar3d (xl,yl,x2,y2,l,top[on/ofr]) - рисуется прямоугольный брусок, и передней гранью на диагонали x1,y1 - x2,y2, глубиной, обусловленной параметром 1. Параметр top[on/off] определяет, имеет ли брусок перекрытие сверху или нет;
Arc (x,y:integer;alfl,alf2,r:word)- рисуется дуга радиусом г с центром в точке x,y от угла alfl до угла alf2. Угол задается в градусах;
Circle (x,y:integer;r:word) - рисуется окружность радиуса r, с центром в точке x,y;

Работа с цветом:

Setlinestyle (stil, pattern, tolsch) - устанавливает тип линии (сплошная, пунктирная, штрих- пунктирная, штриховая) и ее толщину;
Setfillstyle (obr,c:word) - устанавливает образец заполнения для процедур

Bar, Bar3d, Floodfill. Значение параметра obr равное 0 - цветом фона, 1 - сплошное заполнение цветом, указанным параметром c, координаты x,y должны лежать внутри заполняемой области;
Setcolor (c:word) - устанавливает цвет рисования;
Setbkcolor (c:word) - устанавливает цвет фона;
Floodfill (x,y:integer;c:word) - заполняет замкнутые области, ограниченные линией цвета c, в соответствии с образцом, установленным процедурой

Текст:
OutTextXY (x,y:integer; text:string) – выводит текст text, начиная с позиции (x,y);
SetTextStyle (Font, Direction, CharSize: word) – устанавливает стиль текста.

Пример 29. Используя графические примитивы, нарисовать домик

```
uses crt, graph;
var y:real;
  gd,gm,x:integer;
begin
clrscr;
gd:=detect; initgraph(gd,gm,'c:\tp\bgi');
setfillstyle(1,red);
bar(120,140,520,340);
setfillstyle(1,9);
bar(220,190,420,290);
setcolor(red);
line(120,140,320,40);
line(320,40,520,140);
repeat until keypressed;
closegraph;
end.
```

11.2. Построение графика функции

Для того чтобы график соответствовал математическому построению, строить график будем относительно центра экрана. Напоминаем, что начало координат находится в верхнем левом углу экрана. Поэтому при записи координат это необходимо учесть. Кроме того, учесть, что мы искусственно переносим начало координат в центр экрана. Так как коэффициенты функции, а следовательно, и значение y могут оказаться вещественными величинами, а координаты не могут быть вещественными, при построении значения y необходимо привести к целому типу, используя одну из функций приведения типов. График будем строить как множество точек.

При задании интервала построения может оказаться, что интервал очень мал или велик по сравнению с размером экрана. Точно также значения функции могут оказаться или очень малы или очень велики. В

этом случае желательно ввести в программу растягивающие или сжимающие коэффициенты. Заметим также, что для более плотного расположения точек графика, желательно изменять координату x с меньшим шагом, чем 1, для чего можно использовать цикл с предусловием.

Решение данной задачи удобно проводить в следующем порядке:

1. Определить границы значений аргумента, в пределах которых будет строиться график X_{min} - нижняя граница; X_{max} - верхняя граница.
2. Определить предельные значения функции. Эти значения должны быть оценочными.
3. Задать границы графического окна, в пределах которых будет рисоваться график [Xg_{min}, Xg_{max}], [Yg_{min}, Yg_{max}]. Поскольку в графических координатах вертикальная ось направлена вниз, то $Yg_{min} > Yg_{max}$.

Т.о., мы имеем две системы координат: (X,Y) - математическая, (Xg,Yg) - графические координаты. Формула, связывающая графические и математические координаты:

$$Xg = Xg_{\min} + \left[\frac{Xg_{\max} - Xg_{\min}}{X_{\max} - X_{\min}} (X - X_{\min}) \right]$$

$$Yg = Yg_{\min} + \left[\frac{Yg_{\max} - Yg_{\min}}{Y_{\max} - Y_{\min}} (Y - Y_{\min}) \right]$$

График строится в виде последовательности точек с математическими координатами: $X_i = X_{min} + i \cdot h$; $Y_i = F(X_i)$.

Шаг выбирается минимально возможным: $h = \frac{X_{\max} - X_{\min}}{Xg_{\max} - Xg_{\min}}$

Пример 30. Построить график функции y=sin(x), x принадлежит [0;2π].
$X_{min} = 0$, $X_{max} = 2\pi$, $Y_{min} = -1$, $Y_{max} = 1$.
Выберем следующие границы графического окна: $Xg_{min} = 10$, $Xg_{max} = 200$, $Yg_{min} = 140$, $Yg_{max} = 40$.
График строится в виде последовательности точек с математическими координатами

$$X_i = X_{min} + i \cdot h; \; Y_i = \sin(X_i).$$

Приведенные выше формулы перевода математических координат в графические примут вид:

$$Xg = 10 + \left[\frac{200 - 10}{2\pi} X \right] = 10 + \left[\frac{95}{\pi} \right];$$

$$Yg = 140 + \left[\frac{40 - 140}{2} (Y + 1) \right] = 90 - [50Y].$$

Вместе с графиком функции строятся оси координат: ось X имеет координату Y_g=90, ось Y - координату X_g=10;

```
uses crt, graph;
var x:real;
  gd,gm,xg,yg,i:integer;
begin
clrscr;                                  {очистка экрана}
gd:=detect; initgraph(gd,gm,'c:\tp\bgi');  {инициализация графики}
setcolor(white);
setbkcolor(black);
line(10,90,200,90);                      {построение оси X}
line(10,20,10,160);                      {построение оси Y}
x:=0;
for i:=0 to 190 do                       {построение графика}
 begin                                   {точечным методом}
    xg:=10+round(95/pi*x);
    yg:=90-round(50*sin(x));
    putpixel(xg,yg,yellow);
    x:=x+pi/95;
 end;
outtextxy(15,30,'Y');                    {подписи осей и графика}
outtextxy(205,90,'X');
outtextxy(130,40,'Y=SIN(X)');
closegraph;                              {закрытие графического режима}
repeat until keypressed;                 {задержка экрана}
end.
```

Задание 31. Построить график функции y=cos(x), x принадлежит [-2π;2π].

11.3. Анимация

Имитировать движение объекта по экрану можно различными способами.

Наиболее простой способ - рисование объекта, затем очистка графического экрана, и вновь рисование с изменившимися координатами. Это способ применять можно только в качестве демонстрационного примера, так как мигание экрана, особенно у машин с малым быстродействием, очень большое и это неблагоприятно влияет на зрение, да и эффект от такой модели движения невелик,

Более эффективно применение способа перекрашивания объекта в цвет фона экрана. Здесь при большом быстродействии ЭВМ мигание почти не заметно, но необходимо учесть, что построение происходит достаточно медленно и поэтому нежелательно в качестве объектов использовать сложные фигуры, состоящие из большого количества элементов.

Пример 31. Создать на экране движение маленькой окружности.

```
uses crt, graph;
const rad=5; maxx=640; maxy=480; color_border=6; color_fon=0;
var dx,dy,x,y,gd,gm:integer;
```

```
                    f:boolean;

procedure ris(x,y:word;color:word);      {процедура рисования окружности}
begin                                    {заданного цвета}
setcolor(color);
circle(x,y,rad);
end;

begin
gd:=detect; initgraph(gd,gm,'c:\tp\bgi');  {инициализация графики}
randomize;
dx:=1; dy:=1; x:=random(maxx-2*rad)+rad; y:=random(maxy-2*rad)+rad;
repeat                                   {рисование окружности }
   begin
    x:=x+dx; y:=y+dy;
    if (x>=(maxx-rad)) or (x<=(0+rad)) then dx:=-dx;
    if (y>=(maxy-rad)) or (y<=(0+rad)) then dy:=-dy;
     ris(x,y,color_border);
     delay(100);
     ris(x,y,color_fon);
   end;
until keypressed;
closegraph;
end.
```

Задание 31. Используя программу примера 28:

- Увеличьте радиус окружности;
- Уменьшите скорость движения окружности по экрану;
- Измените цвет фона.

Приложение 1. Примерные задачи на экзамен по информатике

1. Составить логическую функцию определения, является ли данное число простым и найти все простые числа, предшествующие данному.
2. Дано число в двоичной системе счисления. Перевести его в десятичную.
3. Составить программу, определяющую, в каком из данных двух чисел больше цифр.
4. Найти площадь поверхности треугольной пирамиды, зная ее ребра (функция определения площади треугольника по формуле Герона).
5. Сформировать случайным образом два массива натуральных чисел и определить, в каком больше чисел, кратных 5.
6. Сформировать 2 массива чисел, определить индексы максимальных элементов и поменять их местами.
7. Дана целочисленная квадратная матрица порядка N. Найти среднее арифметическое элементов, расположенных над побочной диагональю.
8. Дана действительная матрица размером NxM, N>=3, M>=3. Поменять местами два произвольных столбца, номера которых заданы.
9. Дана действительная квадратная матрица порядка N. Подсчитать число нулевых элементов в строке с максимальной суммой элементов.
10. Дана действительная квадратная матрица порядка N. Найти минимум и максимум и поменять их местами.
11. Дано предложение. Убрать из него слова, длина которых меньше 4 букв.
12. Дана строка. Убрать из него все слова-перевертыши (слова, типа «ОХОХО»).
13. Дано предложение. Удалить из него все лишние пробелы (не должно быть более одного пробела между словами).
14. В строке между словами вставить вместо пробела запятую и пробел.
15. Составить программу преобразования натуральных чисел, записанных в римской нумерации, в десятичную систему счисления.
16. Дана строка. Подсчитать количество букв k в последнем ее слове.
17. Написать программу, которая напечатает в порядке убывания все цифры, не входящие в данное натуральное десятичное число (для вводимого числа использовать тип longint).
18. Дан текст на русском языке. Напечатать в алфавитном порядке все согласные буквы, которые не входят ни в одно слово.
19. Написать программу, позволяющую определить дату, которая наступит через x дней после текущей даты.
20. В столовой предлагается N комплексных обедов, состоящих из Q блюд. Известна стоимость каждого блюда. Сколько стоит самый дешевый и самый дорогой обед?

Приложение 2. Задачи для самостоятельного выполнения

Вариант 1

1. Дано натуральное число:
   - найти произведение цифр числа;
   - верно ли, что число начинается и заканчивается одной и той же цифрой?
2. Дан массив целых чисел (n=10), заполненный случайным образом числами из промежутка [-40, 30].
   - Удалить из него все элементы, которые состоят из одинаковых цифр
   - Вставить число k перед всеми элементами, в которых есть цифра 1 (k вводить с клавиатуры).
3. Дана последовательность слов. Напечатать все слова, предварительно выполнив преобразования их по правилу: заменить во всех словах первую букву заглавной.
4. Дан текстовый файл, в котором хранятся данные об учениках нескольких школ: фамилия, имя отчество, адрес (улица, дом, квартира), школа и класс. Вывести на экран фамилию, имя и адрес тех учеников, кто учится в данной школе в старших классах (номер школы вводить с клавиатуры).

Вариант 2

1. Дано натуральное число:
   - найти сумму цифр числа;
   - верно ли, что число заканчивается на нечетную цифру?
2. Дан массив целых чисел (n=10), заполненный случайным образом числами из промежутка [-20, 50].
   - Удалить из него все элементы, в которых есть цифра 5
   - Вставить число k перед всеми элементами, заканчивающимися на данную цифру (k вводить с клавиатуры).
3. Дана последовательность слов. Напечатать все слова, предварительно выполнив преобразования их по правилу: в слове (словах) наибольшей длины удалить последнюю букву.
4. Дан текстовый файл, в котором хранятся данные о расписании поездов: номер поезда, название (откуда - куда, например, Киров - Москва), время прибытия на станцию, время отправления (часы, минуты). Будем считать, что все поезда приходят каждый день. По данному времени определить, какие из поездов стоят сейчас на станции (время вводить с клавиатуры).

Вариант 3

1. Дано натуральное число:
   - найти количество цифр числа;
   - верно ли, что число начинается на четную цифру?
2. Дан двумерный массив размером 8x7, заполненный случайным образом.
   - Заменить все элементы первых трех столбцов на их квадраты.
   - Поменять местами столбцы и строки.
3. Дана последовательность слов. Напечатать все слова, предварительно выполнив преобразования их по правилу: заменить в каждом слове первую встреченную букву «а» на «о».
4. Дан массив данных о работающих на фабрике: фамилия, имя отчество, адрес (улица, дом, квартира) и дата поступления на работу (месяц, год). Определить, есть ли в списке Ивановы (Иванов, Иванова), если есть, то вывести их адрес.

Литература

1. Информатика: Учеб. пособие для студ. пед. вузов / А.В. Могилев, Н.И. Пак, Е.К. Хеннер; Под ред. Е.К. Хеннера. – 2-е изд., стер. – М.: Изд. Центр «Академия». 2001. – 816 с.
2. Семакин И.Г., Шестаков А.П. Основы программирования: Учебник. – М.: Мастерство; НМЦ СПО; Высшая школа, 2001. -432 с.
3. Фаронов В.В. Турбо Паскаль 7.0. Начальный курс. Учебное пособие. – М.: «Нолидж», 1997. – 616 с.

Содержание