МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Воронежский государственный технический университет»

Кафедра высшей математики
и физико-математического моделирования

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ ПАСКАЛЬ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по дисциплинам «Информатика» и «Практикум по информационным технологиям» для студентов направлений 14.03.01 «Ядерная энергетика и теплофизика», 16.03.01 «Техническая физика», 21.03.01 «Нефтегазовое дело», 22.03.02 «Металлургия» и 28.03.01 «Нанотехнологии и микросистемная техника» очной и очно-заочной форм обучения

Воронеж 2021

УДК 004.4:519.671(07)
ББК 32.97я7

Составители:
С. А. Кострюков, В. В. Пешков, Г. Е. Шунин

Основы программирования на языке Паскаль: методические указания к выполнению лабораторных работ по дисциплинам «Информатика» и «Практикум по информационным технологиям» для студентов направлений 14.03.01 «Ядерная энергетика и теплофизика», 16.03.01 «Техническая физика», 21.03.01 «Нефтегазовое дело», 22.03.02 «Металлургия» и 28.03.01 «Нанотехнологии и микросистемная техника» очной и очно-заочной форм обучения / ФГБОУ ВО «Воронежский государственный технический университет»; сост.: С. А. Кострюков, В. В. Пешков, Г. Е. Шунин. – Воронеж: Изд-во ВГТУ, 2021. – 41 с.

В методических указаниях кратко рассмотрены основные приемы создания простейших программ на языке Паскаль, присутствует большое число разобранных программ.

Предназначены для студентов очной и очно-заочной форм обучения.

Методические указания подготовлены в электронном виде и содержатся в файле МУ-Паскаль.pdf.

Ил. 3. Библиогр.: 5 назв.

УДК 004.4:519.671(07)
ББК 32.97я7

Рецензент – И. М. Пашуева, канд. физ.-мат. наук, доц. кафедры высшей математики и физико-математического моделирования ВГТУ

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

ВВЕДЕНИЕ

Несмотря на большое разнообразие и доступность современных математических программ, способных осуществлять решение широкого спектра задач, как в аналитическом, так и в численном виде, владение навыками алгоритмизации и программирования по-прежнему является необходимым для студентов любой технической специальности. Практически все современные математические пакеты, такие, как MathCAD, Maple, Mathematica и т.д., содержат встроенные средства программирования, использующие разветвления, циклы, подпрограммы и др. Для получения студентами навыков использования этих средств и освоения основных приемов программирования наилучшим образом подходит язык программирования высокого уровня Паскаль.

Язык программирования Паскаль был разработан Н. Виртом в 1968-1970 гг. и поначалу предназначался для обучения программированию, однако затем, с появлением Турбо Паскаля фирмы Borland International, получил широкое распространение благодаря наглядности программ, удобству и легкости при изучении. Он послужил основой для разработки других языков программирования (Ада, Модула-2), на его основе создана популярная RAD-среда Delphi.

PascalABC.NET – это система программирования и язык Pascal нового поколения для платформы Microsoft .NET. Язык PascalABC.NET содержит все основные элементы современных языков программирования: модули, классы, перегрузку операций, интерфейсы, исключения, обобщенные классы, сборку мусора, и т.д. Платформа Microsoft .NET обеспечивает язык PascalABC.NET огромным количеством стандартных библиотек и позволяет легко сочетать его с другими .NET-языками: C#, Visual Basic.NET, управляемый C++, Oxygene и др.

Система PascalABC.NET включает в себя также простую интегрированную среду, ориентированную на эффективное обучение современному программированию. Она обеспечивает подсветку синтаксиса, подсказку по коду, форматирование текста программы по запросу, и т.д.

Структура программы на Паскале

Любая программа (или программная единица – процедура или функция) на Паскале состоит из трех основных частей:

1. Раздел заголовка (объявления программной единицы);
2. Разделы описаний;
3. Раздел операторов (тело программы).

В разделе заголовка содержится одна-единственная строчка, которая указывает компилятору, что он имеет дело именно с программой (процедурой или функцией) и, более того, с программой под определенным именем. Для программ эта срока начинается с зарезервированного слова **Program**, после которого следует собственно имя программы. В конце обязательно ставится точка с запятой. Раздел заголовка для основной программы является необязательным.

В разделах описаний должны содержаться описания всех идентификаторов, используемых в разделе операторов или нижележащих разделах описаний. Первым должен идти раздел описания используемых модулей – ключевое слово **Uses**, за которым следует список имен модулей. Модуль PABCSystem называется системным и автоматически подключается к любой программе, создаваемой с помощью PascalABC.NET, указывать его в разделе **Uses** не нужно.

Прочие разделы описаний могут следовать друг за другом в любом порядке и встречаться в разделе описаний сколько угодно раз. Единственное ограничение – идентификаторы, которые используются для определения других идентификаторов, должны описываться раньше. В простейшей программе из всех подразделов раздела описания обычно присутствует только раздел описания переменных **Var**.

Тело программы (раздел исполняемых операторов) содержит собственно программный код, отвечающий за реализацию алгоритма. При этом тело программы обязательно оформляется так называемыми *операторными скобками* – словами **Begin**, **End**. Все операторы, реализующие ваш алгоритм, должны помещаться между ними.

Все разделы программы, кроме раздела операторов, могут отсутствовать, если в них нет нужды.

Таким образом, структура программы имеет вид:

Program <*имя программы*>; {*например: program First;*}
Uses <*список модулей*>; {*например: uses GraphABC;*}
Label <*список меток*>; {*например: Label Home;*}
Const <*имя константы*>=<*значение*>; {*например: const n=7;*}
Type <*имя типа*>=<*описание типа*>; {*например:*
type mass=array[1..3, 1..5] of real;}
Var <*имя переменной*>:<*тип данных*>; {*например: var x: real;*}
Раздел описания процедур и функций. Каждая процедура должна начинаться с ключевого слова **Procedure**, а каждая функция – со слова **Function**.

Begin
<*операторы программы*>
End.

Простейший ввод-вывод данных

Ввод и вывод данных в Паскале полностью реализованы с помощью стандартных процедур.

Для ввода данных с клавиатуры служат процедуры **Read** и **ReadLn**. Примеры:

```
Read(a);
ReadLn(x,y,z);
```

При вводе нескольких значений одной процедурой их разделяют пробелом, пробелами или нажатием клавиши <*Enter*>.

При вводе с клавиатуры используют обычно **ReadLn**, а процедура **Read** предназначена для работы с внешними файлами. Особенность процедуры **ReadLn** в том, что она вызывает **Read** для ввода указанных переменных, а затем, игнорируя остаток введенной строки, переходит к началу следующей строки (*Ln* – сокращение от *Line – строка*).

Для вывода данных на дисплей применяются процедуры **Write** и **WriteLn**. Примеры:

```
Write(a);
WriteLn(x, '    ', y); {между x и y выводится несколько
                        пробелов, взятых в апострофы}
Write('Максимальное значение: ', max);
```

При использовании **WriteLn** после вывода указанных в скобках величин происходит переход на новую строку, чтобы следующий вывод данных шел с новой строки. **Write** не заканчивает строку, и следующая выводимая величина будет печататься в той же строке.

При выводе данных допустимо использование *шаблона вывода*. После выводимой величины ставится двоеточие и указывается число позиций, которые отводятся под нее в текущей строке экрана. Для вещественных чисел также можно указать число позиций под дробную часть числа:

```
WriteLn(x:9:6); { отводится 9 знаков, из них 6 после точки}
```

Обычно при вводе с клавиатуры исходных данных программы предварительно выводят на экран подсказку – что именно пользователь должен ввести. Примеры:

```
Write('Введите x: '); ReadLn(x);
Write('Введите число элементов массива (от 1 до 10):');
ReadLn(n);
```

Основные стандартные функции модуля PABCSystem

Сведения о стандартных типах данных рассматриваемой реализации Паскаля – языка PascalABC.NET – в данных методических указаниях не приводятся, вы можете их найти в материалах лекций, учебниках и справочной системе.

Математические функции (тип результата вещественный)

Abs(x)	$\lvert x \rvert$	**Cos(x)**	$\cos x$
Sqr(x)	x^2	**Sin(x)**	$\sin x$
Sqrt(x)	$\sqrt{x}$	**Tan(x)**	$\mathrm{tg}\, x$
Exp(x)	e^x	**ArcCos(x)**	$\arccos x$
Ln(x)	$\ln x$	**ArcSin(x)**	$\arcsin x$
Log10(x)	$\lg x$	**ArcTan(x)**	$\mathrm{arctg}\, x$

Log2(x)	$\log_2 x$	**Cosh(x)**	ch x
Frac(x)	дробная часть x	**Sinh(x)**	sh x
Int(x)	целая часть x	**Tanh(x)**	th x
Power(x,y)	x^y		

Математические функции (тип результата целый)

Ceil(x)	наименьшее целое >= x
Floor(x)	наибольшее целое <= x
Round(x)	округляет x до ближайшего целого, если x находится посередине между двумя целыми, то округляет к ближайшему четному (банковское округление): Round(2.5) = 2, Round(3.5) = 4
Trunc(x)	усекает x до целого, т.е. находит целую часть x
Sign(x)	–1, 0 или +1 в зависимости от знака числа x

Общие подпрограммы

Процедура **Dec(i)** – уменьшает целое *i* на 1
Процедура **Dec(i,n)** – уменьшает целое *i* на *n*
Процедура **Inc(i)** – увеличивает целое *i* на 1
Процедура **Inc(i,n)** – увеличивает целое *i* на *n*
Функция **Ord(a)** – порядковый номер значения *a*, например, код символа *a*
Функция **Pred(n)** – предшествующее *n* значение
Функция **Succ(n)** – следующее за *n* значение

Стандартные константы

E = 2.718281828459045 – константа *e*
Pi = 3.141592653589793 – число π

Выражения

Выражение – это формальное правило для вычисления некоторого значения. Выражения строятся из операндов, знаков операций и круглых скобок.

Операндами могут быть константы, переменные (в том числе элементы массивов, поля записей, и т.п.), вызовы функций, и др.

Операции – действия по получению новых значений из значений операндов. Большинство операций являются *бинарными*, т.е. определенными для двух операндов. В этом случае знак операции ставится между ними. Примеры: **`a + b`**, **`n div 2`**, **`x > y`**. *Унарные* операции определены для одного операнда, здесь знак операции ставится слева. Примеры: **`–x`** (унарный минус), **`not a`**.

Очередность выполнения операций в выражении определяется их *приоритетами*. Первыми выполняются те операции, чей приоритет выше. Если приоритеты операций равны, то операции выполняются слева направо. Если этот порядок нужно изменить, используются круглые скобки, тогда часть выражения в скобках будет вычислена первой.

Основные операции, определенные в языке Паскаль, и их приоритеты приведены в таблице.

Операция		Приоритет	Категория
@	вычисление адреса переменной	Первый (наивысший)	
not	инверсия		
*	умножение	Второй	Операции типа умножения
/	деление		
div	деление нацело		
mod	остаток от деления нацело		
and	конъюнкция		
shl	логический сдвиг влево		
shr	логический сдвиг вправо		
+ –	сложение, вычитание, унарный плюс, унарный минус	Третий	Операции типа сложения
or	дизъюнкция		
xor	строгая дизъюнкция		
= <> < > <= >=	сравнения	Четвертый (низший)	Операции отношения
in	проверка принадлежности к множеству		

Выражения записываются в виде линейных последовательностей символов (без подстрочных и надстрочных символов, «многоэтажных» дробей и т.д.), что позволяет вводить их в компьютер, последовательно нажимая на соответствующие клавиши клавиатуры. Различают выражения *арифметические*, *логические* и *строковые*.

Арифметические выражения служат для определения одного числового значения. Например, **(1+Sin(x))/2**.

Логические выражения содержат некоторые условия, которые могут выполняться или не выполняться. Таким образом, логическое выражение может принимать только два значения – «False» или «True» («Ложь» или «Истина»). Пример: логическое выражение **x*x + y*y < r*r**, определяющее, лежит ли точка с координатами (*x*, *y*) внутри окружности радиусом *r* с центром в начале координат.

Строковые выражения – выражения, значениями которых являются наборы символов. В строковые выражения могут входить символьные и строковые константы, символьные и строковые переменные, строковые функции, разделенные знаками операции сцепки. Например, **A + B** означает присоединение строки **B** к концу строки **A**. Если **A='куст '**, а **B='зеленый'**, то выражение **A + B** будет иметь значение **'куст зеленый'**.

В PascalABC.NET к строкам применима операция +=:

```
s+=s1;  { то же, что и  s:=s+s1;}
```

Строка может складываться с числом, при этом число предварительно преобразуется к строковому представлению:

```
s:='Длина: '+15;  { то же, что и  s:='Длина: 15';}
```

Над строками и целыми определена операция `*`: `s*n` и `n*s` дает строку, образованную из строки `s`, повторенной *n* раз:

```
s:=4*'ab';  { то же, что и  s:='abababab';}
```

Операция **SizeOf**(*имя типа*) возвращает для этого типа его размер в байтах.

Лабораторная работа №1
Программы разветвляющейся структуры

2 часа

Для программирования разветвлений в программах на языке Turbo Pascal используются два оператора:

1) *Условный оператор* (разветвление на две ветви):

```
If <условие> Then <оператор1>
Else <оператор2>
```

Вторая ветвь (т.е. **Else <оператор2>**) может отсутствовать.

Например:

```
If x>y Then z:=x Else z:=y;
If a<b Then a:=0;
```

Если требуется после слов **Then** или **Else** использовать несколько операторов, то их необходимо объединить в один составной оператор с помощью операторных скобок **Begin** и **End**:

```
If x>y Then Begin a:=x; b:=y End
Else Begin a:=y; b:=x End
```

При использовании сложных логических выражений, включающих операции **and**, **or**, **not**, операции отношения необходимо заключать в скобки, например:

```
If (a>b) and ((x>y) or (x>z)) Then ...
```

Порядок выполнения операций в выражении определяется их приоритетами. Среди логических операций сначала выполняется операция **Not** (наивысший приоритет), затем **And**, последними – **Or** и **Xor** (низший приоритет). Для изменения этого порядка нужно использовать скобки.

В PascalABC.NET для программирования разветвлений также может использоваться условная операция:

условие ? *выражение1* : *выражение2* {первая форма}

Если условие выполняется, то результатом является значение выражения1, иначе значение выражения2. Например:

```
min := a<b ? a : b;
```

Кроме того, имеется вторая форма условной операции:

If *условие* **Then** *выражение1* **Else** *выражение2*

Например:

```
min := If a<b Then a Else b;
```

2) *Оператор варианта* (оператор множественного выбора) обеспечивает разветвление на произвольное число ветвей:

```
Case <выражение> of
<список значений 1> : <оператор 1>;
<список значений 2> : <оператор 2>;
...
<список значений N> : <оператор N>;
Else <оператор>
End
```

Здесь также ветвь **Else** может отсутствовать.

Выражение после **Case** должно быть дискретного типа, совместимого со значениями из списков.

Например:

```
Case ErrorCode of
1:       WriteLn('Файл не найден');
2..4,6:  WriteLn('Ошибка открытия файла: ',
         ErrorCode);
5:       WriteLn('Недопустимое имя файла');
End
```

Практические задания

1. Составить программу, определяющую, попала ли точка с координатами (x, y) внутрь заштрихованной области (рис. 1).

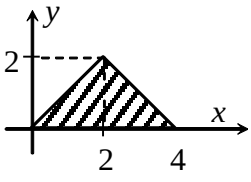

Рис. 1

2. С клавиатуры вводятся три целых числа a, b, c. Изучить и запустить программу, которая выводит их на экран в порядке возрастания (использовать наименьшее количество условных операторов).

```
Program ThreeNumbers;
Var a,b,c,x: integer;
Begin
  Write('a='); ReadLn(a);
  Write('b='); ReadLn(b);
  Write('c='); ReadLn(c);
```

```
  {Упорядочим сначала два числа a, b – должно быть a < b}
  If a>b Then Begin x:=a; a:=b; b:=x End;
  {a и b упорядочены, теперь может быть c < a, c > b или a < c < b}
  If c<a Then WriteLn(c,' ',a,' ',b)
  Else If c>b Then WriteLn(a,' ',b,' ',c)
       Else WriteLn(a,' ',c,' ',b);
End.
```

3. Изучить и запустить программу, которая вычисляет сумму, разность, произведение или частное двух чисел в зависимости от знака операции +, –, *, / (т.е. простейший калькулятор). Добавить в программу операцию возведения *x* в степень *y*.

```
Program Calc;
Var x,y,r:real;
c:char;
Begin
  Write('Операнд 1: '); ReadLn(x);
  Write('Операция : '); ReadLn(c);
  Write('Операнд 2: '); ReadLn(y);
  Case c of
  '+': r:=x+y;
  '-': r:=x-y;
  '*': r:=x*y;
  '/': If y<>0 Then r:=x/y
       Else
       Begin WriteLn('Деление на 0');Exit End;
  Else Begin
         WriteLn('Неверная операция');
         Exit
       End;
  End;
  WriteLn(x:6:3,' ',c,' ',y:6:3,' = ', r:9:6);
End.
```

4. Составить программу, определяющую, попала ли точка с координатами (x, y) внутрь заштрихованной области (рис. 2).

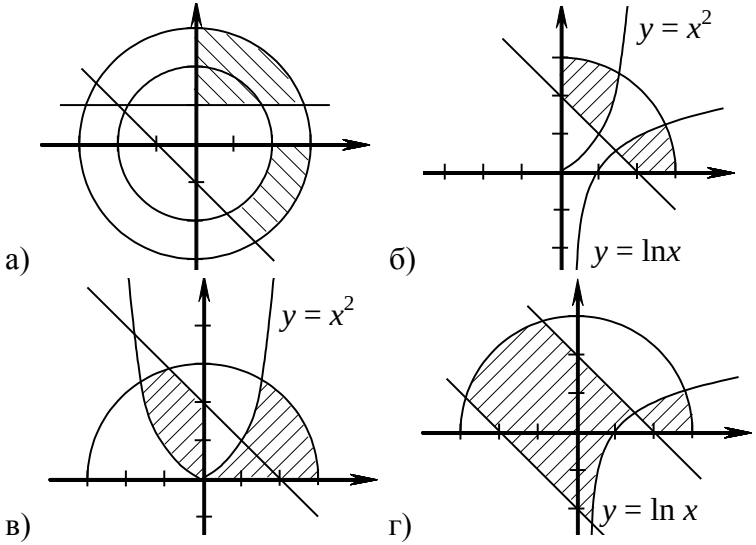

Рис. 2

5. Составить программу, определяющую, можно ли построить треугольник со сторонами a, b, c, и если да, то определяющую его площадь и форму:

а) равносторонний, равнобедренный или разносторонний;

б) прямоугольный, остроугольный или тупоугольный.

6. Составить программу, определяющую площадь и форму треугольника, заданного двумя сторонами и углом между ними.

7. Составить программу вычисления действительных корней квадратного уравнения $ax^2 + bx + c = 0$.

8. На клавиатуре вслепую нажимается клавиша. Составить программу, определяющую тип введенного символа – цифра, прописная буква, строчная буква, спецсимвол (использовать оператор варианта).

9. Используя оператор варианта, составить программу вычисления площадей геометрических фигур – круга, квадрата, прямоугольника, прямоугольного треугольника, произвольного треугольника.

Лабораторная работа №2
Цикл с параметром

2 часа

Наиболее часто в программах на Паскале используется оператор цикла с параметром (оператор безусловного цикла):

```
For i:=N1 to N2 do < тело цикла >;
```

где **i** – параметр цикла, который должен быть дискретного типа (целого, символьного и т.д.); **N1** – начальное значение параметра цикла; **N2** – конечное значение параметра цикла.

Безусловный цикл выполняется заданное число раз. Чтобы прервать выполнение досрочно, необходимо увеличить **i** до конечного значения (**i:=N2**), либо использовать процедуру **Break**, которая прерывает выполнение циклов. Процедура **Continue** начинает новую итерацию цикла, даже если предыдущая не была завершена.

Также может быть использован другой вид оператора **For**, в котором происходит уменьшение значения параметра:

```
For i:=N2 downto N1 do < тело цикла >;
```

Практические задания

1. Изучить и запустить программу, вычисляющую сумму $\sum_{i=1}^{n}\frac{1}{i!}$. Определить максимальное значение *n*, при котором факториал (переменная **f** типа **integer**) вычисляется верно.

```
Var f,n,i: integer;
    s: real;
Begin
  s:=0; f:=1;
  Write('Введите N: '); ReadLn(N);
  For i:=1 to n do begin
    f:=f*i;
    s:=s+1/f
  End;
  WriteLn('s=',s:9:6);
End.
```

2. Изучить и запустить программу, вычисляющую значение выражения $\sum_{k=1}^{N}(-1)^k \frac{x^{2k}}{(2k)!}$, где N – любое целое число.

```
Var s,f,ch,x: real;
    k,n: integer;
Begin
  Write('Введите число n: '); ReadLn(n);
  Write('Введите число x: '); ReadLn(x);
  s:=0; f:=1; ch:=1;  {начальные присваивания}
  For k:=1 to n do {цикл вычисления суммы}
  Begin
    ch:=-ch*x*x; {вычисление нового числителя}
    f:=f*(2*k-1)*(2*k);{вычисление нового факториала}
    s:=s+ch/f
  End;
  WriteLn('s=', s);
End.
```

3. Составить программы, вычисляющие значения выражений:

а) $\sum_{i=1}^{100}\frac{1}{i^2}$; б) $\sum_{i=1}^{N}\frac{1}{\sqrt[3]{i^2}}$; в) $\sum_{k=1}^{N}\frac{\sin^k x}{k!}$ г); $\sum_{i=1}^{5}\frac{x^{i+1}}{(3i)!}$;

д) $\prod_{i=1}^{5}\frac{i+2}{i^2+1}$; е) $\prod_{k=1}^{N}\left(1+\frac{\ln(kx)}{k}\right)$; ж) $\sum_{i=1}^{10}\sum_{j=i}^{10}\frac{1}{i+j^2}$; з) $\sum_{i=1}^{100}\sum_{j=1}^{i}\frac{1}{2j+i}$;

и) $\sin x+\sin^2 x+...+\sin^n x$; к) $\frac{1}{\sin 1}+\frac{1}{\sin 1+\sin 2}+...+\frac{1}{\sin 1+...+\sin n}$.

4. Изучить и запустить программу вычисления значений функции на интервале $x_1 \le x \le x_2$ с шагом h и найти ее минимальное значение и среднее арифметическое этих значений:

$$y=\begin{cases}\sin^2|x|, & \text{если } x>a;\\ \cos x^2, & \text{если } x\le a.\end{cases}$$

```
Program Tabulation; {табулирование – вывод таблицы значений}
Var x,y,a, x1, x2, h, xmin, ymin, SA: real;
    n,i: integer;
Begin
WriteLn('Введите начало и конец интервала,
число a и шаг h');
ReadLn(x1, x2, a, h);
If (x2>x1) and ((x2-x1)>2*h) then { если x2 > x1 и}
Begin                  { на интервале не менее трех точек}
  x:=x1;               { инициализация }
  n:=Round((x2-x1)/h); {кол-во точек равно n+1 }
  If(x>a) Then y:=Sqr(Sin(Abs(x)))
  Else y:=Cos(x*x);          {вычисление функции }
  WriteLn('x=', x:7:3,'  y=', y:9:4); {вывод x1 и y1}
  xmin:=x; ymin:=y; SA:=y; { присваивание начальных }
                             { значений перед циклом }
  For i:=1 to n do           { выполняется цикл }
  Begin
    x:=x1+i*h;
    If(x>a) Then y:=Sqr(Sin(Abs(x)))
    Else y:=Cos(x*x);        {вычисление функции }
    WriteLn('x=', x:7:3,'  y=', y:9:4); {вывод x и y}
    If y<ymin Then
    Begin
      ymin:= y; xmin:= x;    {корректируем ymin, xmin}
    End;
    SA:=SA+y;          {накопление суммы значений функции}
  End;                 { конец цикла }
  WriteLn('Минимальное значение y =',ymin:9:4,'
при x=', xmin:7:3);
  WriteLn('Среднее арифметическое: ',SA/(n+1));
End
Else WriteLn('Неверно заданы x1 и x2');
End.
```

5. Составить программу, которая для целых чисел k от 1 до N выводит на экран k, k^2, $\sqrt{k}$, $k!$

6. Составить программу, находящую сумму всех двузначных чисел: а) которые делятся на 7 и при этом не делятся на 5; б) которые делятся на 3 и не делятся на 4; в) у которых число единиц делится нацело на число десятков.

7. Составить программу, выводящую на экран все простые числа в интервале от 1 до 100 (простым называется число, которое делится нацело только на 1 и на себя).

8. Составить программы, вычисляющие значения выражений:

а) $\sum_{k=1}^{10}\frac{(-1)^{k+1}}{k(k+1)}$; б) $\sum_{k=1}^{n}\frac{1}{\sqrt[k]{k^5}}$; в) $\sum_{k=1}^{n}\frac{k!}{\frac{1}{2}+\frac{1}{3}+...+\frac{1}{k+1}}$; г) $\sum_{i=1}^{n}\sum_{j=1}^{i}\frac{(i-j+1)^i}{(i+j-1)^j}$;

д) $\prod_{i=1}^{10}\frac{i^2}{i^2+2i+3}$; е) $\prod_{i=1}^{10}\left(2-\frac{1}{i!}\right)$; ж) $\prod_{k=1}^{n}(2+\sqrt{kn})$; з) $\prod_{k=2}^{10}\frac{\log_k 2}{(k-1)!}$;

9. Составить программу вычисления значений функций на интервале $x_1 \le x \le x_2$ с шагом h и найти минимальное, максимальное и среднее арифметическое этих значений:

а) $y=\frac{2^{-x}+2^{-2x}}{2^x+2^{2x}}$; б) $y=\begin{cases}10^x, & x<0;\\ \cos(\pi x/2), & x\ge 0.\end{cases}$

Лабораторная работа №3
Циклы с условиями

2 часа

Помимо уже рассмотренного оператора цикла с параметром **For**, в языке Паскаль имеется два оператора цикла с условиями. Они используются в случаях, когда заранее невозможно определить, сколько раз должно выполниться тело цикла.

1) Оператор цикла с постусловием Repeat ... Until. В этом операторе проверка условия выхода из цикла осуществляется после каждого выполнения тела цикла, поэтому тело цикла обязательно выполнится хотя бы один раз:

```
Repeat
< тело цикла >
Until < условие >;
```

Цикл выполняется до тех пор, пока условие не станет истинным. Например, при задании длины массива проверка может осуществляться следующим образом:

```
Repeat
  Write('Введите длину массива N (N<10): ');
  ReadLn(N)
Until (N>0) and (N<10);
```

Цикл выполняется до тех пор, пока пользователь не введет положительное число, меньшее 10.

Здесь в теле цикла можно использовать несколько операторов, не заключая их в операторные скобки **Begin** ... **End**.

2) Оператор цикла с предусловием While ... Do. В этом цикле проверка условия проводится до начала выполнения тела цикла:

```
While < условие > Do < тело цикла >;
```

Цикл выполняется, пока условие истинно. Как только условие становится ложным, выполнение цикла завершается. Если условие ложно с самого начала, тело цикла не выполнится ни разу. Например, та же проверка вводимой длины массива может быть задана так:

```
N:=0;
While (N<=0) or (N>=10) Do
Begin
  Write('Введите длину массива N: ');
  ReadLn(N)
End;
```

Обратите внимание, что до входа в цикл переменной **N** должно быть присвоено значение, не удовлетворяющее условию!

Процедуры **Break** и **Continue** могут использоваться в условных циклах так же, как и в цикле **For**.

Практические задания

1. Вычислить с помощью рекуррентной формулы сумму ряда с заданной точностью ε (использовать оператор цикла с постусловием):

$$\sum_{n=0}^{\infty}(-1)^n\frac{x^{2n}}{(2n)!}.$$

Решение. Каждый член данного ряда требует вычисления только двух функций – степенной и факториала. Поэтому в рекуррентную формулу целесообразно включить весь член ряда

$$u_n=(-1)^n\frac{x^{2n}}{(2n)!}.$$

Рекуррентная формула позволяет получить новый член ряда u_{n+1} через предыдущий u_n (при этом нужно найти u_0):

$$u_{n+1}=R_n u_n,\quad u_0=1.$$

Определим множитель R_n

$$R_n=\frac{u_{n+1}}{u_n}=\frac{(-1)^{n+1}x^{2(n+1)}}{(2(n+1))!}\frac{(2n)!}{(-1)^n x^{2n}}=-\frac{x^2}{(2n+1)(2n+2)}.$$

Итак, сумму ряда можно найти по следующей схеме. Новое значение суммы S равно предыдущему значению (тоже S) плюс новый член ряда:

$$S=S+u_{n+1},$$

где

$$u_{n+1} = -\frac{x^2}{(2n+1)(2n+2)} u_n .$$

Суммирование заканчивается, когда $|u_{n+1}| < \varepsilon$.

```
Var s,u,x,eps: real;{s – сумма, u – член ряда, eps – точность}
    n: integer; {x – аргумент функции, n – номер члена ряда}
Begin
  WriteLn('Введите аргумент x и точность eps');
  ReadLn(x, eps);
  u:=1;  s:=u;  n:=0; { инициализация переменных}
  Repeat
    u:=-u*x*x/(2*n+1)/(2*n+2); {рекуррентная формула}
    s:=s+u; {суммирование ряда}
    n:=n+1; {счетчик}
  Until abs(u)<eps;
  WriteLn('Сумма ряда равна ', s);
  WriteLn('Для сравнения cos(x)=', cos(x));
End.
```

2. Составить программу решения уравнения $\sin\sqrt{x} = 0$ на отрезке $[a; b]$ с точностью ε методом деления отрезка пополам.

Решение. Если на отрезке $[a; b]$ имеется корень уравнения $f(x) = 0$, то график функции $y = f(x)$ пересекает ось Ox между точками a и b (рис. 3). В этом случае значения функции $y = f(x)$ на концах отрезка $[a; b]$ имеют разные знаки:

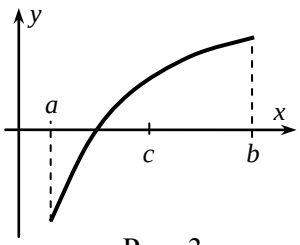

Рис. 3

$$f(a) \cdot f(b) < 0.$$

Это неравенство – условие существования корня уравнения $f(x) = 0$ на отрезке $[a; b]$. Если оно не выполняется, отрезок $[a; b]$ задан неверно.

Далее отрезок $[a; b]$ делится пополам точкой $c = (a + b)/2$, и одна из половин отрезка (на которой нет корня) отбрасывается.

Для этого одна из границ переносится в точку c (если условие $f(a)\cdot f(c)<0$ истинно, то это точка b, если ложно – точка a).

Процесс деления отрезка завершается, когда его длина становится меньше точности ε.

```
Var a,b,c,eps: real;
Begin
  Write('Задайте границы отрезка a и b: ');
  ReadLn(a,b);
  Write('Задайте точность решения: ');
  ReadLn(eps);
  If sin(sqrt(a))*sin(sqrt(b))<0 Then
  Begin
    While abs(b-a)>eps Do
    Begin
      c:=(a+b)/2;
      If sin(sqrt(a))*sin(sqrt(c))<0 Then b:=c
      Else a:=c
    End;
    WriteLn('x=',(a+b)/2);
    WriteLn('f(x)=',sin(sqrt((a+b)/2)))
  End
  Else WriteLn('Неверный отрезок');
End.
```

3. Вычислить суммы рядов с точностью ε = 0,0001, не используя рекуррентные формулы. Использовать цикл с предусловием (а, в, д), с постусловием (б, г, е):

а) $\sum\limits_{n=0}^{\infty}\frac{1}{(2n+1)^2}=\frac{\pi^2}{8}$; б) $\sum\limits_{n=1}^{\infty}\frac{1}{n^4}=\frac{\pi^4}{90}$; в) $\sum\limits_{n=2}^{\infty}\frac{1}{\sqrt{n}}\ln\frac{n+1}{n-1}$;

г) $S=\sum\limits_{n=1}^{\infty}\left(n\cdot\arcsin\frac{1}{n}\right)^n$; д) $\sum\limits_{n=2}^{\infty}\left(\frac{n-1}{n+1}\right)^{n^2}$; е) $\sum\limits_{n=2}^{\infty}\frac{(-1)^n}{n-\ln n}$.

4. Вычислить суммы рядов с точностью ε = 0,0001. Использовать, где это целесообразно, рекуррентные формулы для всего члена ряда или для его части.

а) $\sum_{n=1}^{\infty}(-1)^{n+1}\frac{2^n}{n!}$; б) $\sum_{n=1}^{\infty}\frac{(-1)^n 2^{-n+1}}{n(n+1)(n+2)}$; в) $\sum_{n=1}^{\infty}\frac{1}{n!(x+3)^n}$;

г) $\sum_{n=1}^{\infty}\frac{(-1)^{n-1}\ln(n+1)}{n(x+1)^n}$; д) $\sum_{n=1}^{\infty}\frac{n(n+2)}{(2n)!}$; е) $\sum_{n=2}^{\infty}\frac{(-1)^n}{3^{n-1}\cdot\sqrt[n]{x^2}}$.

5. С клавиатуры вводится действительное число $x\in(0;\ \pi)$ и точность ε. Проверить равенства и определить, сколько слагаемых было просуммировано:

а) $\sin x+\frac{\sin 3x}{3}+\cdots+\frac{\sin(2n-1)x}{2n-1}+\cdots=\frac{\pi}{4}$;

б) $\frac{\sin 2x}{1\cdot 3}+\frac{2\sin 4x}{3\cdot 5}+\frac{3\sin 6x}{5\cdot 7}+\cdots+\frac{n\sin 2nx}{(2n-1)\cdot(2n+1)}+\ldots=\frac{\pi}{8}\cdot\cos x$.

6. Проверить равенство. С клавиатуры ввести вещественное число x и точность ε. Определить, сколько сомножителей было найдено.

а) $x\cdot\left(1-\frac{x^2}{\pi^2}\right)\cdot\left(1-\frac{x^2}{2^2\pi^2}\right)\cdot\left(1-\frac{x^2}{3^2\pi^2}\right)\cdot\ldots=\sin x$;

б) $\left(1-\frac{4x^2}{\pi^2}\right)\cdot\left(1-\frac{4x^2}{3^2\pi^2}\right)\cdot\left(1-\frac{4x^2}{5^2\pi^2}\right)\cdot\ldots=\cos x$.

Лабораторная работа №4
Работа с массивами и строками

4 часа

1) Тип «массив»

Массив – последовательность однотипных элементов, число которых фиксировано и которым присвоено одно имя. Положение элемента в массиве однозначно определяется его индексами (одним, в случае одномерного массива, или несколькими, если массив многомерный). Иногда многомерные массивы называют таблицами или матрицами. Описание:

```
Var <имя>: Array[<тип1>, <тип2>, …, <типN>]
of <базовый тип>;
```

Здесь ***<тип1>, <тип2>, …, <типN>*** – типы индексов, их число определяет размерность массива (один – для одномерного массива, два – для двумерного, и т.д.). Чаще всего здесь используется тип «диапазон». Базовый тип – тип элементов массива. Примеры:

```
Var A: Array[1..3] of real;
```
{*массив из трех вещественных чисел*}

```
    B: Array[1..10, 1..10] of integer;
```
{*матрица 10×10 из целых чисел*}

```
    C: Array[Char] of integer;
```
{*массив из 256 целых чисел, значения индекса – символы*}

Также массив может быть определен с помощью описания нового типа:

```
Type Vector = Array[1..3] of real;
     Matrix = Array[1..10, 1..10] of integer;
Var A: Vector;
    B: Matrix;
```

Этот способ применяется, если массив нужно передать в подпрограмму в качестве параметра.

В PascalABC.NET при описании можно также задавать инициализацию массива значениями:

```
Var A: Array[1..3] of real := (1,2,3);
```

В PascalABC.NET процедуры **Write** и **WriteLn** производят вывод массива, заключая элементы в квадратные скобки и разделяя их запятыми:

```
WriteLn(A); { Результат: [1,2,3] }
```

Единственная операция над массивом в целом – присваивание, например, **B:=A;** (при этом массивы **A** и **B** должны быть определены одинаково). Все остальные операции над массивами производятся поэлементно, чаще всего в циклах. Обращение к элементам массивов производится путем указания в квадратных скобках значений индексов: **A[1], B[2,3], B[i,j],** и т.п. Допустима другая форма: **B[2][3]**.

2) Тип «строка»

Строковый тип расширяет понятие символьного массива (**Array[…] of char**), предоставляя новые возможности. Описание:

```
Var <имя>: String[<макс. длина>];
```

Если максимальная длина не указана явно, то по умолчанию берется значение 255. Примеры:

```
Var s: String[80];
    mess: String;
```

При создании в программе строки с максимальной длиной N символов под нее отводится $N+1$ байт памяти. Элементы с номерами от 1 до N предназначены для хранения символов, а элемент с номером 0 – для хранения динамической (т.е. текущей) длины строки. Определить текущую длину строки **S** можно с помощью функций **Length(s)** или **Ord(s[0])**.

Строковые константы, как и символьные, заключаются в апострофы:

```
s:='Пример строки';
```

Если при присваивании максимальная длина строковой переменной оказывается меньше длины строковой константы, то лишние символы отбрасываются.

Операции над строками:

а) конкатенация (т.е. слияние строк): две строки соединяются в одну:

```
s:='Пример ' + 'строки';
```

б) сравнение по правилам:

– более короткая строка меньше, чем более длинная;

– если длины строк равны, то попарно сравниваются коды первых символов, если они равны, то вторых, и т.д.

Доступ к отдельным символам строки производится так же, как и к элементам массивов: `s[3]:='a'`;

И Турбо Паскаль, и PascalABC.NET содержат большое число стандартных функций для работы со строками, познакомиться с которыми можно в справочной системе среды программирования.

Практические задания

1. Изучить и запустить программу, осуществляющую ввод массива с клавиатуры, сортировку его по убыванию и вывод на экран. С ее помощью составить программу, выполняющую следующие действия над вещественным одномерным массивом $A(n)$:

а) поиск и вывод на экран минимального и максимального элементов с указанием номера;

б) перестановку элементов массива (1-го с последним, 2-го с предпоследним, и т.д.);

в) сортировку элементов массива по возрастанию методом пузырька;

г) замену каждого из элементов массива a_i на разность $a_i - a_{ср}$, где $a_{ср}$ – среднее арифметическое элементов массива.

```
Var A: Array[1..100] of real;
    i,j,n: integer;
    z: real;

Begin
  {Ввод массива A(n)}
  Write('Введите длину массива (n<100):');
  ReadLn(n);
  WriteLn('Введите элементы массива:');
```

```
  For i:=1 to n do
  Begin
    Write('A[', i, ']=');
    ReadLn(A[i])
  End;
  {Вывод массива в одну строку}
  WriteLn('Массив A:');
  For i:=1 to n do Write(A[i]:6:3, ' ');
  WriteLn;
  {Сортировка}
  For i:=1 to n-1 do
    For j:=i+1 to n do
      If A[i]<A[j] Then
      Begin
        z:=A[i]; A[i]:=A[j]; A[j]:=z
      End;
  {Вывод отсортированного массива}
  WriteLn('Отсортированный массив:');
  For i:=1 to n do Write(A[i]:6:3, ' ');
  WriteLn
End.
```

2. Изучить и запустить программу, осуществляющую ввод матрицы A с клавиатуры, перестановку указанных строк и вывод матрицы на экран в виде таблицы. С ее помощью составить программу, выполняющую следующие действия над вещественным двумерным массивом $A(m\times n)$, $m \neq n$:

а) поиск и вывод на экран минимального и максимального элементов матрицы с указанием их позиции;

б) подсчет количества положительных и отрицательных элементов матрицы A;

в) транспонирование матрицы A;

г) вычисление и печать результатов перемножения матриц A и A^{T}: $B = AA^{\mathrm{T}}$, $C = A^{\mathrm{T}}A$.

д) перестановку столбцов матрицы A, чтобы максимальный элемент оказался в первом столбце, и перестановку строк, чтобы он оказался в левом верхнем углу матрицы.

```
Var A: Array [1..10,1..10] of real;
    i,j,n,m,m1,m2: integer;
    z: real;
Begin
  {Ввод матрицы A}
  Write('Введите число строк (m<10): ');
  ReadLn(m);
  Write('Введите число столбцов (n<10): ');
  ReadLn(n);
  For i:=1 to m do
    For j:=1 to n do
    Begin
      Write('A[', i, ',', j, ']=');
      ReadLn(a[i,j])
    End;
  {Вывод матрицы в виде таблицы}
  WriteLn('Исходная матрица:');
  For i:=1 to m do
  Begin
    For j:=1 to n do Write(a[i,j], '  ');
    WriteLn
  End;
  {Ввод номеров переставляемых строк}
  Write('Введите номера переставляемых строк: ');
  ReadLn(m1,m2);
  If (m1>m) or (m2>m) or (m1=m2) Then
  Begin WriteLn('Неверные номера'); Exit End;
  {Перестановка}
  For j:=1 to n do
  Begin
    z:=a[m1,j]; a[m1,j]:=a[m2,j]; a[m2,j]:=z
  End;
  {Вывод матрицы в виде таблицы}
  WriteLn('Матрица после перестановки:');
  For i:=1 to m do
  Begin
    For j:=1 to n do Write(a[i,j], '  ');
    WriteLn
  End
End.
```

3. Изучить и запустить программу, подсчитывающую, сколько раз встречается каждый символ в задаваемой с клавиатуры строке.

```
Var s: string;
    i, max: word;
    Num: array [char] of byte;
    c: char;
Begin
  If sizeof(char)=1 Then max:=255
  Else max:=65535;
  Writeln('Введите строку:'); Readln(s);
  For c:=chr(0) to chr(max) do Num[c]:=0;
  For i:=1 to length(s) do
    Num[s[i]]:=Num[s[i]]+1;
  For c:=chr(0) to chr(max) do
    If Num[c]>0 Then
    Writeln('Символов ''',c,''' - ',Num[c],
' шт. Код ',ord(c));
End.
```

4. Составить программу, которая в массиве $A(15)$ подсчитывает число отрицательных и число неотрицательных элементов и формирует из них два массива, в один из которых войдут неотрицательные, а в другой – отрицательные элементы.

5. В программе на Паскале с клавиатуры ввести массив A из N целых чисел ($N > 2$). Найти и удалить из него минимальный элемент, сдвигая последующие к началу массива.

6. В программе на Паскале с клавиатуры ввести вещественный массив A из 10 чисел и число x. Заменить все элементы, меньшие x, на среднее арифметическое элементов массива.

7. В программе на Паскале с клавиатуры ввести массив A из N целых чисел, где N – четное число. Поменять местами:

а) наибольший и наименьший элементы;

б) 1-й и 2-й элементы, 3-й и 4-й, и т.д.

в) 1-й и 3-й элементы, 2-й и 4-й, и т.д.

г) первую и вторую половины массива.

8. Составить программу, которая в одномерном массиве длиной n находит второй максимальный по модулю элемент.

9. В программе на Турбо Паскале с клавиатуры ввести матрицу целых чисел $A(n \times n)$. Вывести на экран:

а) главную и побочную диагонали матрицы;

б) строку с наименьшим элементом матрицы;

в) столбец с наибольшим элементом матрицы;

г) сколько элементов в каждой строке матрицы больше заданного числа M;

д) сколько элементов в каждом столбце матрицы меньше заданного числа M;

е) среднее арифметическое каждого столбца матрицы.

10. Составить программу, осуществляющую ввод с клавиатуры вещественных матриц $A(3 \times 3)$ и $B(3 \times 3)$ и выполняющую следующие действия:

а) сложение матриц: $C = A + B$;

б) перемножение матриц: $C = AB$; $D = BA$;

в) вычисление скалярного произведения указанной строки матрицы A и того же столбца матрицы B.

11. Составить программу, которая по введенной с клавиатуры вещественной матрице $A(n \times n)$ формирует одномерный массив B, элементами которых являются суммы элементов каждой строки $\boldsymbol{A}$, деленные на максимальный элемент $\boldsymbol{A}$.

12. Составить программу, которая во введенной с клавиатуры строке удаляет лишние пробелы между словами, оставляя один.

13. Составить программу, которая в строке, введенной пользователем, находит число слов, начинающихся и заканчивающихся одной и той же буквой.

14. Составить программу, которая во введенной с клавиатуры строке находит слово наибольшей длины.

Лабораторная работа №5
Процедуры и функции

4 часа

В Паскале существует два типа подпрограмм – процедуры и функции. Описание всех процедур и функций, созданных пользователем, должно присутствовать в разделе подпрограмм. Для использования большинства стандартных подпрограмм необходимо в разделе **Uses** подключить к программе модуль, содержащий эту подпрограмму.

1) Процедуры

Структура подпрограммы, в том числе процедуры, в целом идентична структуре основной программы – заголовок, разделы описаний и раздел операторов, только в отличие от основной программы заголовок здесь не может отсутствовать:

```
Procedure <имя>(<список формальных параметров>);
<разделы описаний>
begin
  <операторы>
end
```

Список формальных параметров представляет собой перечисление имен параметров с указанием их типов. Например, процедура поиска и вывода максимума из двух вещественных чисел может выглядеть так:

```
Procedure Print_max(x,y: real);
Begin
  If x>y Then Writeln(x:9:6)
  Else Writeln(y:9:6)
End
```

Вызов процедуры может производиться из любого места основной программы или других подпрограмм с помощью оператора вызова процедуры. Указывается имя процедуры и список фактических параметров, которые будут подставлены вместо формальных:

```
Print_max(a, b);
```

или

```
Print_max(a+b, 2*c);
```

Между формальными и фактическими параметрами должно быть строгое соответствие по количеству, типу и порядку следования.

Чтобы передать в подпрограмму в качестве параметра массив, строку или другую переменную составного типа, этот тип данных должен быть описан в разделе **Type** с присвоением нового имени. Например, процедура печати первых *N* элементов массива:

```
Type mass = array[1..100] of real;
Var a: mass;
    n: integer;
Procedure print_mass(z: mass; n: integer);
Var i: integer;
Begin
  For i:=1 to n do Write(z[i]:6:2,'  ');
  Writeln
End
```

2) Функции

Описание функции похоже на описание процедуры – заголовок, разделы описаний, раздел операторов. Отличие заключается в том, что функция имеет результат, тип которого указывается после круглых скобок (его имя совпадает с именем функции):

```
Function <имя>(<форм. параметры>): <тип ре-
                                    зультата>;
<разделы описаний>
Begin
  <операторы>
End
```

Чтобы результат функции был вычислен, внутри нее должен быть хотя бы один оператор присваивания, в котором имя функции стоит слева. Например, функция вычисления длины вектора имеет вид:

```
Type vector = array[1..3] of real;
Function Len(z: vector): real;
Begin
  Len:=Sqrt(z[1]*z[1]+ z[2]*z[2]+z[2]*z[2])
End;
```

Вызов функции, как стандартной, так и определенной самим программистом, обязательно должен входить в какое-либо выражение в качестве операнда:

```
L:=Len(a);
x:=(1+Cos(t))/2;
Writeln('sin y = ', Sin(y));
```

Обмен данными между основной программой и подпрограммой производится с помощью параметров. Параметры могут быть следующих типов: *параметры-значения*, *параметры-переменные*, *нетипизированные параметры* и *параметры-константы*.

С помощью *параметров-значений* можно передавать данные только в одном направлении – из основной программы в подпрограмму, т.е. через них передаются исходные данные для подпрограммы. При этом в качестве фактических параметров, стоящих в скобках при вызове подпрограммы, могут использоваться любые выражения, тип результата которых совместим с типом соответствующего формального параметра. Эти выражения могут включать переменные, константы, вызовы функций и др.

Если в подпрограмме параметры-значения изменятся в процессе вычислений, то эти изменения никак не отразятся ни на одной переменной той программы, откуда была вызвана подпрограмма. Поэтому для возвращения результатов из подпрограммы в основную программу параметры-значения не подходят. Такой способ передачи параметров называется пере-

дачей *по значению*, поскольку формальному параметру присваивается *значение* фактического параметра. В этом случае формальный параметр будет содержать копию значения, имеющегося в фактическом, и никакое воздействие, производимое внутри подпрограммы на формальные параметры, не отражается на параметрах фактических.

Параметры-переменные позволяют передавать данные в обоих направлениях – и из основной программы в подпрограмму, и обратно. Поэтому с их помощью можно возвращать в основную программу результаты, полученные в подпрограмме. Это возможно потому, что в подпрограмму передается не значение фактического параметра, а ссылка на него, т.е. его адрес в памяти. Поэтому все изменения формального параметра в подпрограмме – это на самом деле изменения фактического параметра, так как в этом случае оба параметра суть одно и то же. Такой способ передачи параметров называется передачей *по ссылке.* В этом случае в качестве фактических параметров можно использовать только переменные (в том числе элементы массивов, поля записей и т.п.), а выражения и константы использовать нельзя. При описании подпрограммы в списка формальных параметров перед параметрами-переменными ставится слово **Var**. Например, заголовок процедуры, возвращающей минимальное и максимальное значения из элементов массива длиной *N*:

```
Type mass = array[1..100] of real;
Var a: mass;
Procedure MinMax(z: mass; n: integer; var min,
max: real);
```

В случае *параметров-констант* перед их именами ставится слово **Const**. При их использовании в подпрограмму также передается ссылка, т.е. адрес параметра в памяти, однако компилятор блокирует все попытки изменить его в подпрограмме, т.е. присвоить ему какое-либо значение.

Массивы в подпрограмму рекомендуется передавать по ссылке, а не по значению – при большом размере массива это

приводит к ненужным расходам оперативной памяти. Если массив не меняется внутри подпрограммы, то его следует передавать как параметр-константу, если меняется – как параметр-переменную.

Если формальный параметр является *нетипизированным* параметром-переменной, то соответствующий ему фактический параметр может представлять собой любую ссылку на переменную, независимо от ее типа. Описание таких параметров выглядит как группа параметров с ключевым словом **Var** впереди, за которыми не следует указание типа, например:

```
Function Fun(var First, Second): Boolean;
```

Если некая переменная объявлена в разделе **Var** основной программы, то ее называют *глобальной* переменной. Она существует от места описания до конца программы (эту область называют областью видимости данной переменной). Таким образом, она видна и может быть использована и во всех подпрограммах этой программы.

Если некая переменная объявлена в разделе **Var** какой-либо подпрограммы, то она считается *локальной* переменной этой подпрограммы. Она существует от места описания до конца этой подпрограммы, т.е. только в этой подпрограмме.

Если некоторое имя определено в скобках в заголовке подпрограммы, то это формальный параметр этой подпрограммы.

Встретив в тексте подпрограммы какой-либо идентификатор, компилятор начинает искать его описание сначала в разделе **Var** этой подпрограммы, затем в списке формальных параметров, и если не находит, то продолжает поиск во внешнем по отношению к этой подпрограмме блоке, пока не доберется до раздела **Var** основной программы. Таким образом, если имя некоторой глобальной переменной совпадет с именем локальной переменной подпрограммы, то в этой подпрограмме можно будет обращаться только к локальной переменной, а глобальная будет недоступна. Говорят, что в этих случаях локальное имя закрывает (маскирует) глобальное.

Считается, что в подпрограммах использовать глобальные переменные следует с осторожностью, так как это может привести к ошибкам.

Практические задания

1. Изучить и запустить программу вычисления площади треугольника, заданного тремя сторонами *a*, *b*, *c*, по формуле Герона, в которой проверка существования треугольника и вычисление площади реализованы в виде функций.

```
Program Geron;
Var a,b,c,s: real;

Function Exist(x,y,z: real): boolean;
Begin
  If (x<=0) or (y<=0) or (z<=0) or
     (x+y<z) or (x+z<y) or (y+z<x) Then
    exist:=false
  Else exist:=true;
End;

Function Square(x,y,z: real): real;
Var p: real;
Begin
  p:=(x+y+z)/2;
  Square:=Sqrt(p*(p-x)*(p-y)*(p-z))
End;

Begin
  Writeln('Введите длины сторон a,b,c:');
  Readln(a,b,c);
  If Exist(a,b,c) Then
  Begin
    s:=Square(a,b,c);
    Writeln('Площадь S=',s:9:6)
  End
  Else Writeln('Треугольник не существует');
End.
```

2. Изучить и запустить программу вычисления длины вектора, заданного в виде массива из 3 чисел, в которой вычисление длины вектора реализовано в виде функции. С ее помощью составить программу обработки векторов, заданных в виде массива из 3 чисел, выполняющую следующие действия: а) вычисление скалярного произведения двух векторов; б) вычисление угла между двумя векторами; в) вычисление векторного произведения двух векторов.

```
Program Vectors;
type vect=array[1..3] of real;
var a,b: vect;
    z: real;

Function Len(x:vect):real;
Begin
  len:=Sqrt(x[1]*x[1]+x[2]*x[2]+x[3]*x[3])
End;

Begin
  Writeln('Введите компоненты вектора');
  Readln(a[1],a[2],a[3]);
  z:=Len(a);
  Writeln('Длина вектора a: ',z:6:2);
End.
```

3. Изучить и запустить программу, осуществляющую ввод квадратных матриц A и B и вывод их суммы $C = A + B$ (ввод, вывод и сложение матриц реализованы в виде процедур). С ее помощью составить программу обработки матриц, выполняющую следующие действия:

а) транспонирование матрицы A;

б) вычисление произведения двух матриц $C = AB$;

в) вычисление следа матрицы (т.е. суммы элементов главной диагонали);

г) поиск максимального и минимального из элементов матрицы и их позиции;

д) обмен содержимого элементов матрицы A симметричным образом относительно главной (или побочной) диагонали.

```
Program Matrix;
type matr=array[1..10,1..10] of real;
var a,b,c: matr;
    n: integer;

Procedure Inp(var x:matr; n:integer; t:char);
Var i,j:integer;
Begin
  For i:=1 to n do
    For j:=1 to n do
    Begin
      Write(t,'[', i, ',', j, ']=');
      Readln(x[i,j])
    End
End;

Procedure Outp(const x:matr; n:integer);
Var i,j:integer;
Begin
  For i:=1 to n do
  Begin
    For j:=1 to n do Write('  ',x[i,j]:6:2);
    Writeln
  End
End;

Procedure Sum(const x,y:matr; n:integer; var
s:matr);
Var i,j:integer;
Begin
  For i:=1 to n do
    For j:=1 to n do s[i,j]:=x[i,j]+y[i,j]
End;

Begin
  Writeln('Введите размерность матриц n: ');
  Readln(n);
  Writeln('Введите матрицу A');
```

```
  Inp(a,n,'A');
  Writeln('Матрица A:');
  Outp(a,n);
  Writeln('Введите матрицу B:');
  Inp(b,n,'B');
  Writeln('Матрица B: ');
  Outp(b,n);
  Sum(a,b,n,c);
  Writeln('Матрица C=A+B:');
  Outp(c,n);
  Readln
End.
```

4. Используя функцию, составить программу решения квадратного уравнения $ax^2 + bx + c = 0$.

5. Используя функции, составить программу решения уравнения $f(x) = 0$ на отрезке $[a, b]$ с заданной точностью ε методом половинного деления.

6. Составить программу решения системы трех линейных уравнений с тремя неизвестными методом Крамера. Вычисление определителя 3-го порядка оформить в виде функции.

7. Треугольник задан координатами вершин на плоскости. Используя функции, составить программу вычисления длин сторон треугольника, углов между сторонами и площади треугольника.

8. Дан целочисленный массив A длиной 10. Используя функции, составить программу, определяющую:

а) среднее арифметическое элементов массива;

б) номер максимального элемента;

в) количество четных элементов;

г) сколько элементов меньше заданного числа M.

Массив должен быть передан в функции как параметр.

9. Используя функцию, составить программу, определяющую, перпендикулярны ли два вектора $\vec{a}$ и $\vec{b}$, сонаправлены,

противоположно направлены, или ни одна из этих ситуаций не имеет места. Векторы заданы тройками чисел (компонент).

10. Используя процедуры, составить программу обработки одномерного целочисленного массива, которая:

а) находит количество четных и нечетных элементов;

б) формирует из одного исходного массива два, из четных и нечетных элементов соответственно;

в) удаляет из массива идущие подряд одинаковые элементы, оставляя один.

11. Используя процедуры, составить программу обработки вещественных квадратных матриц, которая:

а) находит и печатает строку с наибольшим элементом;

б) формирует и печатает массив из максимальных элементов каждой строки;

в) формирует и печатает массивы из элементов главной и побочной диагоналей матрицы соответственно;

г) делит каждый элемент матрицы на максимальный по модулю элемент.

Матрица должна быть передана в процедуры как параметр.

12. Используя процедуры, составить программу обработки строк, которая:

а) печатает самое длинное слово в строке;

б) находит и печатает все слова заданной длины;

в) заменяет в заданной строке все прописные буквы на строчные;

г) удаляет лишние пробелы между словами, оставляя один.

Строка должна быть передана в процедуры как параметр.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Фаронов В. В. Турбо Паскаль 7.0. Начальный курс. Учебное пособие / В. В. Фаронов. – М.: Издательство «ОМД Групп», 2003. – 616 с.

2. Фаронов В. В. Турбо Паскаль 7.0: Учеб. пособие / В. В. Фаронов. – СПб.: Питер, 2007. – 367 с.

3. Программирование на языке Паскаль: задачник / под ред. Усковой О. Ф. – СПб: Питер, 2003. – 336 с.

4. Немнюгин С. А. Turbo Pascal / С. А. Немнюгин. – СПб: Питер, 2000. – 496 с.

5. Осипов А. В. PascalABC.NET: Введение в современное программирование / А. В. Осипов. – Ростов-на-Дону, 2019 – 572 с.

ОГЛАВЛЕНИЕ

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ ПАСКАЛЬ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по дисциплинам «Информатика» и «Практикум по информационным технологиям» для студентов специальностей 14.03.01 «Ядерная энергетика и теплофизика», 16.03.01 «Техническая физика», 21.03.01 «Нефтегазовое дело», 22.03.02 «Металлургия» и 28.03.01 «Нанотехнологии и микросистемная техника» очной и очно-заочной форм обучения

Составители:
Кострюков Сергей Александрович,
Пешков Вадим Вячеславович,
Шунин Геннадий Евгеньевич

Отпечатано в авторской редакции

Подписано к изданию 30.11.2021.
Уч.-изд. л. 2,6

ФГБОУ ВО «Воронежский государственный технический университет»
394026 Воронеж, Московский проспект, 14