МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТРАНСПОРТА

Кафедра "Информационные технологии"

КЕЙЗЕР А.П., РОГАЧЁВА Н.А., ЛИТВИНОВИЧ Т.Н.

ИНФОРМАТИКА

ЯЗЫК ПРОГРАММИРОВАНИЯ ПАСКАЛЬ

Методическое пособие для студентов
дневной формы обучения технических специальностей

Часть 2

Одобрено методической комиссией
факультета управления процессами перевозок
Белорусского государственного
университета транспорта

Гомель 2002

УДК 681.3.06
И 741

Р е ц е н з е н т ы: доцент каф. “Прикладная математика”, к.т.н. С.И. Жогаль
ст. преподаватель каф. “Информационные технологии” А.Б. Зборовская

И 741 Информатика. Язык программирования Паскаль: Методическое пособие для студентов дневной формы обучения технических специальностей / *Кейзер А.П., Рогачева Н.А., ЛитвиновичТ.Н.* – Гомель: БелГУТ, 2002.- 31с.

Данное методическое пособие является продолжением по информатике охватывает следующие разделы: структура типов данных языка Паскаль, оператора цикла, формат описания и использования одномерных и двумерных массивов, описание и использование пользовательских процедур и функций.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Катализатором научно-технического прогресса на транспорте и в народном хозяйстве является широкое использование ЭВМ. Без компьютеров немыслимо развитие производства и науки. Для увеличения объемов выпускаемой продукции и повышения качества необходимо внедрять во все отрасли народного хозяйства автоматизированные и роботизированные комплексы.

Формированию будущего инженера способствует развитие его алгоритмического мышления. Основами алгоритмизации должны обладать инженер-технолог и инженер-конструктор. Даже если инженеру непосредственно не придется программировать при разработке и внедрении новых производственных задач с использованием ЭВМ, то может грамотно поставить задачу математику-программисту, обладая инженерно-алгоритмическим мышлением.

Данное методическое пособие по **информатике** охватывает следующие разделы:

- структура типов данных языка Паскаль, в котором приведена общая схема типов данных и примеры по каждому типу. Даны пояснения по их использованию;
- рассмотрены три основные оператора цикла для выполнения циклических операций и даны пояснения по их работе и особенностям использования;
- формат описания и использования одномерных и двумерных массивов;
- описание и использование пользовательских процедур и функций и на основе обработки массивов даны рекомендации по их использованию.

1 СТРУКТУРА ТИПОВ ДАННЫХ В ЯЗЫКЕ ПАСКАЛЬ

Каждый элемент данных относится к одному из конечного множества типов, допустимых для конкретного языка программирования. Тип - это множество значений, которые могут принимать объекты программы, и совокупность операций, допустимых над этими значениями.

Все типы данных языка Паскаль разделяются на две группы: скалярные (простые) и структурированные (составные).

1.1 Простые типы данных

- **Целочисленные типы (Integer, Byte, Word)** данных представляют собой значения, которые могут использоваться в арифметических выражениях и занимать память от 1 до 4 байт.

Тип	Диапазон	Размер в байтах
Byte	0 ... 255	1
Shortint	-128 ... 127	1
Word	0 ... 65535	2
Longint	-2147483648 ... 2147483647	4

- **Вещественные типы (Real, Double)** данных представляют собой вещественные значения, которые могут использоваться в арифметических выражениях и занимать память от 4 до 10 байт.

Тип	Диапазон	Знач. цифры	Размер в байтах
Real	2.9E-39 …1,7E38	7-8	4
Single	1.5E-45 ... 34E38	15-16	8
Double	5E-324 ... 1.7E308	19-20	10
Extended	1.6E-4951...1.1E4932		

- **Символьный (char)** тип определяется множеством значений кодовой таблицы ASCII. Для переменной символьного типа требуется 1 байт.
- **Логический тип (Boolean)** представлен двумя значениями: (истина) и (ложь). Он широко применяется в логических выражениях и выражениях отношения.

Пользовательские типы данных

В Паскаль существуют типы данных, определяемые пользователем. Это перечислимый тип (когда непосредственно, в разделе описания типов, заранее записываются все значения для переменных этого типа) и интервальный (когда задаются границы диапазона значений для данной переменной).

- **Перечислимый тип** данных задается непосредственно перечислением всех значений, которые может принимать переменная данного типа. При описании отдельные значения указываются через запятую, а весь список заключается в круглые скобки.

  Например: **Var** Season: (winter, spring, summer, autum);
  Temp: (23,24,25,26);

- **Интервальный тип** позволяет задавать две константы, определяющие границы диапазона значений для каждой переменной. Обе константы должны принадлежать одному и тому же стандартному типу (кроме real). *Например:* **Var** S:1..30; Ch:'a'..'f';

1.2 Структурированные типы данных

Составные типы данных определяют упорядоченную совокупность скалярных переменных и характеризуются типом своих компонентов.

К структурированным типам данных в Turbo Pascal относят:

- тип-массив (**array**);
- тип-множество (**set**);
- тип-запись (**record**);
- файловый тип (**file**);
- объектный тип (**object**);
- строковый тип (**string**).

- **Строковый тип**

  Строка - в общем случае это последовательность символов. В строках типа **String** текущая длина строки указывается в нулевом (то есть имеющем индекс 0) элементе строки. Максимальная текущая длина строки может быть не более 255 символов.

- **Тип-массив**

  Массив – это структурированный тип данных, состоящий из фиксированного числа элементов, имеющих один и тот же тип. Элементами массива могут быть данные любого типа, включая структурированный тип.

- **Тип-запись**

  Запись- тип данных, состоящий из фиксированного числа компонентов одного или нескольких типов.

2 ПРОГРАММИРОВАНИЕ ЦИКЛИЧЕСКИХ ВЫЧИСЛЕНИЙ

Вычислить и напечатать значение функции $p5 = sin(a1/b)$, где $a1$=2,8; b изменяется в интервале [2; 5] с шагом Δb = 0,5.

Для вычисления значения *P5* необходимо взять аргументы $a1$=2,8 (постоянное число) и b=2 (начальное значение). Далее необходимо получить новое значение аргумента b = 2 + 0,5 (к начальному значению прибавляется шаг db = 0,5) и для него вычислить и напечатать значение *P5*. Следующий раз *P5* вычисляется при значении b=2,5+0,5. Последний раз значение *P5* вычисляется и печатается при b=5 (конечное значение аргумента).

$a1$ =2,8 b = 2 $p5$=…
$a1$ =2,8 b = 2,5 $p5$=…
$a1$ =2,8 b = 3 $p5$=…
…
$a1$ =2,8 b = 5 $p5$=…

2.1 Оператор цикла WHILE

Структура оператора WHILE

Формат оператора *while*:

while **<условие>** ***do*** **<оператор>;**

где

while, do **–** служебные слова **(пока** [выполняется условие] **делать);**

<условие>–выражение логического типа (например $b <= 5$);

<оператор> - произвольный оператор *Pascal*, который будем называть телом цикла. Оператор может быть простым или составным.

Пример простого оператора: *p5:=sin(a1/b)*

Пример составного оператора:

```
begin
    p5:=sin(a1/b);
    writeln (b, p5);
    b:=b+0.5;
end.
```

Группа простых операторов, заключенная в операторные скобки *begin-end* называется **составным оператором.**

Примечание Пояснение оператора *Writeln* и всех остальных операторов, приведенных в текстах программ, дается в разделе 2.3.

Принцип работы оператора WHILE

1) вычисляется значение логического выражения (проверяется условие);
2) если результатом вычисления значения логического выражения (проверки условия) является ответ “да”, то выполняется простой или составной оператор, составляющий тело цикла и стоящий после служебного слова *do*;
3) далее происходит возврат к пункту 1 и повторяются вышеописанные действия 1, 2;
4) повторение действий 1, 2 продолжается, пока результатом вычисления значения логического выражения (проверки условия) не станет ответ “нет”, после чего управление передается на следующий после цикла оператор программы.

Блок-схемы алгоритма и программа решения задачи с помощью оператора While

Вариант 1

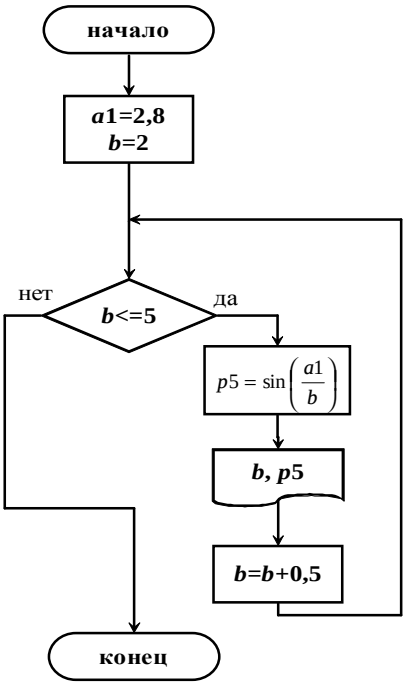


Рисунок 2.1

Вариант 2

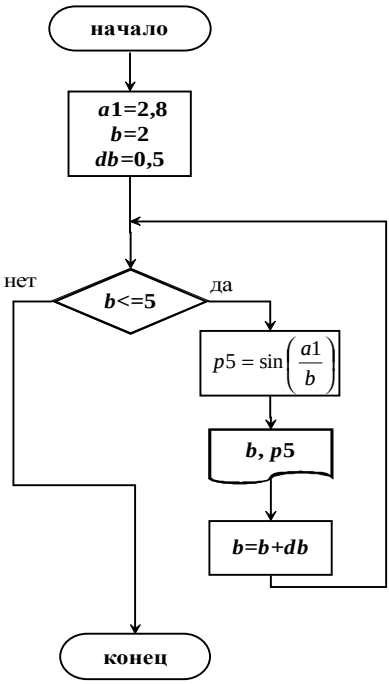


Рисунок 2.2

Алгоритм, приведенный на блок-схеме (рисунок 2.2), обобщает вычисление значения функции на отрезке с шагом *db*. Ниже приведена программа, реализующая алгоритм для рисунка 2.1.

```
Program Primer; {имя программы}
Var {раздел описания переменных}
 a1,b,p5: real; {имена переменных и их тип}
begin {раздел операторов}
 a1=2.8; b=2;
while b<=5 do
  begin
   p5:=sin( a1/b ); {вычисление значения функции}
   writeln(b:8:3,p5:10:3);{вывод значений b и p5}
   b:=b + 0.5;
  end;
end. {конец программы}
```

Рисунок 2.3

2.2 Оператор цикла Repeat. . Until

Структура оператора цикла Repeat. . Until

Формат оператора *repeat-until*:

repeat
< операторы цикла> ;
until **<условие>;**

где

repeat, until **–** служебные слова **(повторять** до тех пор, **пока не** будет выполнено условие);

<операторы цикла> – повторяющаяся в цикле последовательность операторов *TP***;**

<условие> – выражение логического типа.

Принцип действия оператора цикла Repeat ..UNTIL

1. выполняются операторы, расположенные между операторами *REPEAT…UNTIL*, составляющие тело цикла;
2. вычисляется значение логического выражения оператора *UNTIL* (проверяется условие);
3. если результатом вычисления значения логического выражения (проверки условия) является ответ “нет”, то операторы, составляющие тело цикла, выполняются вновь;
4. повторение действий 1, 2 продолжается до тех пор, пока результатом вычисления значения логического выражения (проверки условия) не станет ответ “да”.

Блок-схема алгоритма и программа решения задачи №1 с помощью оператора Repeat-Until

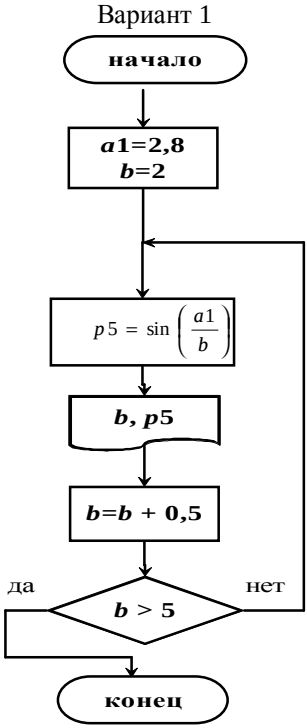


Рисунок 2.4

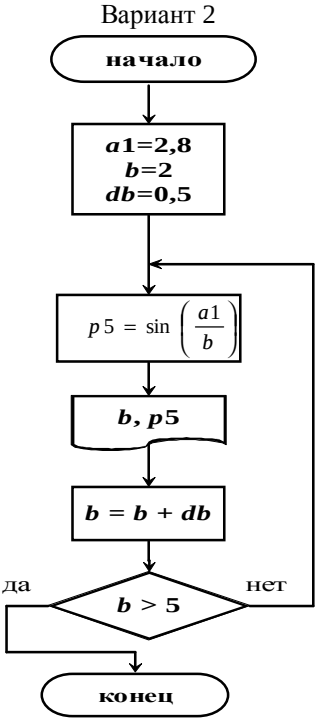


Рисунок 2.5

Блок-схема, представленная на рисунке 2.5 (вариант 2), как и блок-схема (рис.2.2) обобщает вычисление функции *P*5 с шагом *db* с использованием оператора цикла, проверяющего условие выполнение в конце цикла. Приведем текст программы для блок-схемы рис.2.5.

```
Program Primer; {имя программы}
Var a1,b,db,p5: real; {имена переменных и их тип}
begin {раздел операторов}
 a1=2.8;  b=2; db:=0.5;
 repeat
   p5:=sin( a1/b ); {вычисление значения функции}
   writeln(b:8:3,p5:10:3);{вывод значений b и p5}
   b:=b + db;
 until b>5;
end. {конец программы}
```

Рисунок 2.6

2.3 Оператор цикла FOR

Структура оператора FOR

Один из возможных форматов оператора *for*:

***for* <параметр цикла> :=<нач_знач> *to* <*кон_нач*> *do* <оператор>;**

где

***for, to, do* –** служебные слова **(для, до, выполнить**);

<параметр цикла> – параметр цикла; в качестве параметра цикла часто выступает переменная типа *integer* или *byte*;

<нач_знач> – начальное значение параметра цикла - выражение того же типа;

<кон_знач> – конечное значение параметра цикла - выражение того же типа;

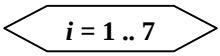
<оператор> –повторяющаяся последовательность операторов *TP***;**

<условие> – выражение логического типа.

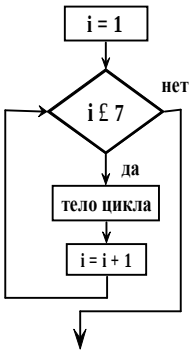
Принцип действия оператора FOR

1. первый раз <оператор>, составляющий тело цикла выполняется при <пар_цикла> : = <нач_знач>;
2. второй раз тело цикла выполняется при <пар_цикла> : = <нач_знач>+ 1;
3. последний раз тело цикла выполняется при пар_цикла> : = <кон_знач>;

Анализ двух блок-схем (рис. 2.7, 2.8) позволяет глубже понять принцип действия оператора *FOR*. Блок,



соответствующий оператору `for i:=1 to 7 do` объединяет 3 действия (i=1; $i \le 7$; $i = i$+1), где



i – вспомогательная переменная, параметр цикла;

i=1 ($i_{нач}$) соответствует первому вычислению функции $P5$ при начальном значении аргумента b = 2 ($b_{нач}$);

i = 2 соответствует 2-му вычислению $P5$ при b = 2,5;

i = 7 ($i_{кон}$) соответствует последнему вычислению функции $P5$ при конечном значении аргумента b=5 ($b_{кон}$).

Количество повторений цикла n можно определить по следующей формуле

$$n = i_{кон} = \frac{b_{кон} - b_{нач}}{db} + 1 = \frac{5-2}{0{,}5} + 1 = 7$$

Блок-схемы алгоритма и программа решения с помощью оператора FOR

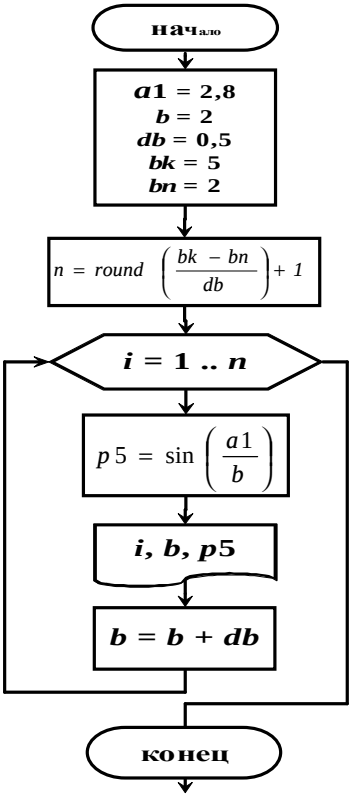


Рисунок 2.7

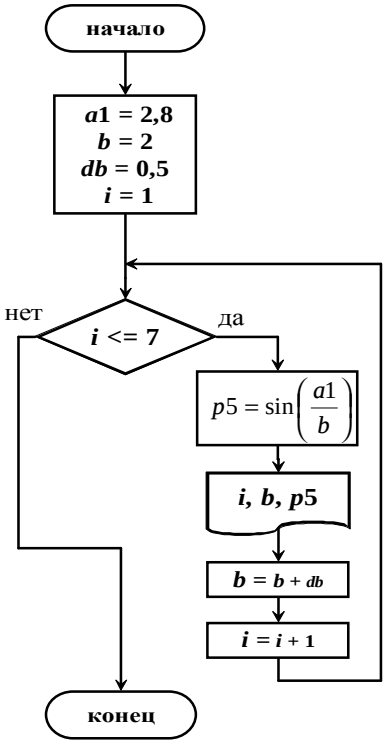


Рисунок 2.8

Приведем программу решения задачи № 1 с помощью оператора *FOR* (рис. 2.7)

```
Program Primer5; {имя программы}
Var {раздел описания  переменных}
       a1, b, dn, p5, bkon, db, ps: real;
       n,I: integer
begin {раздел операторов}
   a1:=2.8; bn:=2;  bk:=5; b:=bn; db:=0.5;
    n:= Round((bk-bn)/db)+1;
    for i:=1 to n do
      begin  p5:=sin( a1/b ); {вычисление значения функции}
             writeln(i:2,b:8:3,p5:10 :3);{вывод значений }
             b:=b + db end
      end.
```

Рисунок 2.9

Функция *ROUND(*<аргумент>*)* округляет до ближайшего целого числа значение аргумента, указанного в скобках. В нашем случае – это вещественный аргумент (*bk* - *bn*)/*db*.

2.4 Практические задания

Задание 1. Вычислить значения функции на некотором диапазоне при заданном шаге изменения аргумента. По исходным данным таблицы составить две блок схемы и две программы на языке *PASCAL*. При расчетах использовать следующие виды циклических вычислений:

1. Цикл с предусловием (проверка условия в начале цикла: оператор *WHILE*).
2. Цикл с постусловием (проверка условия в конце цикла: оператор *REPEAT..UNTIL*).

Результаты вычислений представить в виде таблиц, в которых напечатать значения аргумента и значения функции.

Задание выбирается из таблицы 2.1 по последней цифре номера студента в журнале.

Таблица 2.1

№ п/п	Вычисляемая функция	Диапазон изменения аргумента функции	Шаг аргумента функции
0	$s5 = tg\dfrac{y5}{y5 + p1}$; $p1 = 0{,}8$	$3 \le y5 \le 4{,}5$	$\Delta y5 = 0{,}25$
1	$set1 = \cos^3\dfrac{z}{2y}$; $y = 1{,}3$	$2{,}5 \le z \le 3{,}5$	$\Delta z = 0{,}25$
2	$om1 = \sqrt[5]{\dfrac{s}{s + t4}}$; $t4 = 2{,}9$	$2 \le s \le 8$	$\Delta s = 2$
3	$c = e^{z+w}$; $z = 0{,}8$	$0{,}5 \le w \le 1{,}25$	$\Delta w = 0{,}25$
4	$a2 = \dfrac{t3}{t3 + e^t}$; $t = 1{,}4$	$1{,}5 \le t3 \le 4$	$\Delta t3 = 0{,}5$
5	$del = \sin^2\dfrac{d}{d - p}$; $p = 0{,}8$	$6 \le d \le 18$	$\Delta d = 3$
6	$q = arctg\sqrt{k1}$	$0{,}25 \le k1 \le 1{,}5$	$\Delta k1 = 0{,}125$
7	$LP = \left\lvert \sin^3\dfrac{p3}{p + p3} \right\rvert$; $p3 = 1{,}6$	$3 \le p \le 9$	$\Delta p = 2$
8	$og = b^{a+1,4}$; $a = 0{,}26$	$2 \le b \le 3{,}5$	$\Delta b = 0{,}25$

9	$sm = \sqrt{\frac{b}{2 \cdot c5}}$; $b = 1{,}68$	$2 \le c5 \le 3$	$\Delta c5 = 0{,}25$

Задание 2.

Вычислить значение выражения. По исходным данным таблицы составить блок схему и программу на языке *PASCAL*. При расчетах использовать оператор цикла FOR.

Задание выбирается из таблицы 2.2 по номеру студента в журнале подгруппы.

Таблица 2.2

№ п/п	Задание
1	Дано n . Вычислить $\sum_{i=1}^{n} 2^i$
2	Дано n . Вычислить $(1+\frac{1}{1^2})(1+\frac{1}{2^2})(1+\frac{1}{3^2})\cdot\ldots\cdot(1+\frac{1}{n^2})$
3	Дано n . Вычислить $\sqrt{2+\sqrt{2+\sqrt{2+\ldots+\sqrt{2}}}}$
4	Дано n. Вычислить $\sqrt{n+\sqrt{(n-1)+\sqrt{(n-2)+\ldots+\sqrt{2+\sqrt{1}}}}}$
5	Дано действительное a . Вычислить $\frac{1}{a}+\frac{1}{a\cdot a}+\ldots+\frac{1}{a^a}$
6	Дано: n – натуральное, a – вещественное. Вычислить a^n
7	Дано: a - вещественное, n - натуральное. Вычислить $\frac{1}{a}+\frac{1}{a^2}+\frac{1}{a^4}+\ldots+\frac{1}{a^{2n}}$
8	Даны a - вещественное , n - натуральное. Вычислить $a(a-n)(a-2\cdot n)\ldots(a-n\cdot n)$

9	Вычислить $(1+sin0{,}1)(1+sin0{,}2)...(1+sin1{,}0)$
10	Дано вещественное x, натуральное n. Вычислить $sinx+sinx^2+sinx^3+...+sinx^n$
11	Дано вещественное x . Вычислить $\frac{(x-2)(x-4)(x-8)...(x-64)}{(x-1)(x-3)(x-7)...(x-63)}$
12	Даны x- вещественное , n - натуральное. Вычислить $\sin x+\sin^2 x+...+\sin^n x$
13	Вычислить $\sqrt{3+\sqrt{6+\sqrt{9+...+\sqrt{96+\sqrt{99}}}}}$
14	Вычислить $\sum_{i=1}^{100} \frac{1}{i^2}$
15	Вычислить $\sum_{i=1}^{20} x_i$, где $x_i = (i-1)/3$

3 ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МАССИВОВ

3.1 Описание массивов

Массив – это набор объектов одного типа, у каждого из которых есть индекс (номер). Так, в данном случае массивом является совокупность значений Z_0, Z_1, Z_2, Z_3, Z_4, Z_5. При описании массива указывается общее число входящих в массив элементов и тип этих элементов.

Формат объявления массива:

<имя пер.> **: *array*** **[**<размер массива>**]** ***of*** <тип элементов массива>**;**

Например: Var Z := array[0..5] of real, где

array… of – служебные слова **(массив, из);**

[0 .. 5] – тип-диапазон, определяющий количество элементов массива; левая и правая границы задают номера, соответственно, первого и последнего элементов массива;

real – тип элементов, образующих массив (в данном случае элементы массива вещественные числа).

Доступ к каждому элементу массива в программе осуществляется с помощью его индекса (номера). *Например:* ***Z***[2] : = 5, где 2 – индекс элемента массива ***Z.***

3.2 Обработка одномерных массивов

Рассмотрим пример задачи с обработкой одномерного массива. Ввести массив $Z[0..5]$ и число $P9$. Для каждого Z_i вычислить и напечатать $PT_i = \sqrt[3]{p9 + Z_i}$. Найти сумму значений массива Z и минимальное значение среди элементов массива PT.

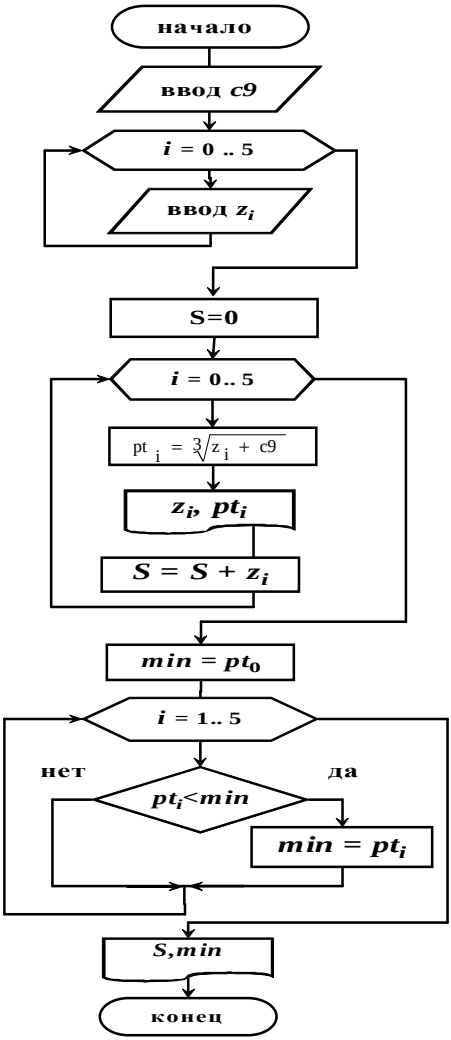


Рисунок 3.1

Программа и блок-схема алгоритма решения задачи

```
Program primer6;
var
  c9,s,min:real;
  i:integer;
  z,pt:array[0..5] of real;
begin
```

```
     readln(c9);
     for i:=0 to 5 do  readln(z[i]);
      s:=0;
     for i=0 to 5 do
       begin
       pt[i]:=exp(1/3*ln(z[i]+c9));
       writeln(z[i]:4:1,pt:10:5);
       s:=s+z[i];
       end;
      min:=pt[0];
      for i:=1 to 5 do if pt[i]< min then min := pt[i];
      writeln(s,min)
end.
```

Рисунок 3.2

3.2.1 Сортировка одномерных массивов

Один из наиболее распространенных процессов обработки данных — сортировка массива.

Так, например, фамилии в телефонном справочнике, банковские картотеки клиентов всегда отсортированы для облегчения доступа к нужной информации. Сортировка — это размещение объектов в определенном порядке. Числа могут размещаться по убыванию или по возрастанию, фамилии — в алфавитном порядке.

Известно несколько алгоритмов сортировки. Мы рассмотрим только один из них — метод сортировки обменом.

Итак, рассмотрим задачу о размещении чисел из таблицы:

50 | 40 | 10 | 20 |…| 30 | 7 |

в порядке неубывания.

Сортировку обменом называют еще методом пузырька. Суть метода состоит в том, что последовательно сравниваются пары соседних элементов массива. Если первый элемент пары оказался больше второго, то они меняются местами и на второе место (как пузырек) «всплывает» больший из двух элементов:

40 | 50 | 10 | 20 |…| 30 | 7 |

Затем выбирается пара, состоящая из 2-го и 3-его элементов массива, сравнение и перестановка повторяются, и на 3-е место всплывает больший элемент из трех. Сравнение с перестановкой повторяются, пока не будет достигнут конец массива, в результате чего самый большой элемент массива «всплывает» и занимает крайнее правое место:

40 | 10 | 20 | 30 |…| 7 | 50 |

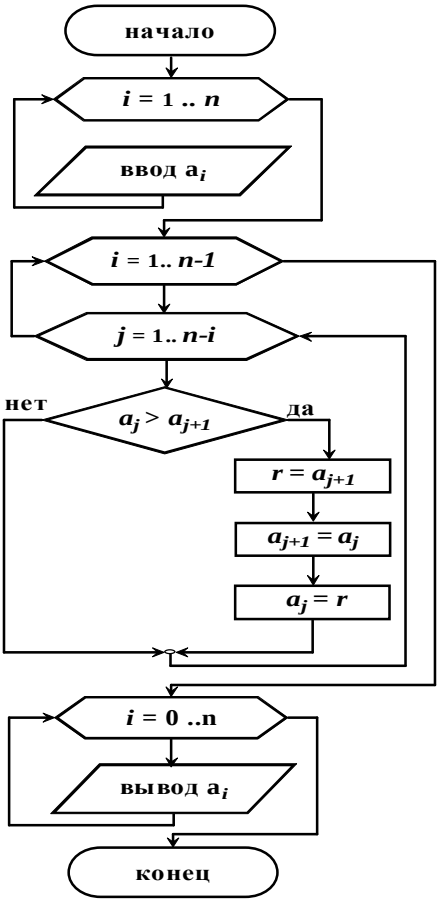


Рисунок 3.3

Такой проход от начала к концу массива составляет один шаг процесса сортировки. В результате выполнения прохода самый большой элемент оказался самым правым элементом массива.

Следующим шагом алгоритма является проход от первого до (n-1)-го элемента. Его результатом будет размещение наибольшего из оставшихся элементов на предпоследнем месте и т. д.

До сих пор речь шла о расположении чисел в порядке их не убывания. А как сделать, чтобы сортировка обеспечивала убывание или не возрастание? Нужно вместо условия A [j] > A [j+1] записать прямо противоположное ему условие A[j] < A[j+1].

Заметим, что возможность появления двух одинаковых чисел не создает каких-либо проблем. В момент сравнения двух одинаковых элементов оба остаются на прежних местах, но затем, постепенно перемещаясь по ряду, они займут свои окончательные положения, оставаясь в соседних позициях.

Блок-схема и программа алгоритма решения задачи представлены на рис.3.3 и 3.4

```
   program sort;
   const n=7;
   a:array[1..n] of real=(50,40,10,20,30,5,7);
   var i,j,k:integer; r:real;
   begin
     for i:=1 to n-1 do {на каждом шаге пузырек "плывет" от
1-ой позиции в (n-i+1)-ю позицию}
      for j:=1 to n-i do
   if a[j]>a[j+l] then  {Сравниваются пары элементов}
     begin
```

```
      r:=a[j+1];           {обмен}
      a[j+1]:=a[j];
      a[j]:=r;
    end;
    WriteLn('Массив отсортирован');
      {Вывод результата сортировки}
    for k:=1 to n do
      Write(a[k]:9:4,'|');
  end.
```

Рисунок 3.4

Так же производится сортировка в случае, если в элементах массива хранятся символы или строки. Такие массивы сортируются в соответствии с кодами символов элементов.

3.3 Обработка двумерных массивов

Двумерные массивы легче всего представить в виде матрицы, где элементы расположены по строкам и столбцам.

Например:

Var M: **array** [1..2,1..3] **of real**; {описание двумерного массива, состоящего из 6-ти вещественных элементов}

Ему будет соответствовать матрица из двух строк и трёх столбцов

M [2 , 3]

индекс строки индекс столбца

M [1, 1]	M [1, 2]	M [1, 3]
M [2, 1]	M [2, 2]	M [2, 3]

Многомерные массивы располагаются в памяти по строкам, т.е. таким образом, что самый правый индекс возрастает самым первым. Для описания массива можно использовать предварительно определенные константы:

```
Const N1= 10;  N2= 20;
Var  Mas : array[1..N1,1..N2] of real;
```

Тогда при использовании программы можно задавать различное число элементов, не изменяя текста программы. Например, при отладке программы можно ограничиться массивом из 6 элементов, а при эксплуатации требуется массив из 200 элементов. В этом случае объявляют массив максимальной размерности, а конкретное число элементов массива запрашивают у пользователя. Приведем фрагмент программы с задаваемым числом элементов в

массиве, а также рассмотрим процедуру ввода элементов двумерного массива.

```
Program Lab5_1;
Const Predel=65;
Var M:array[1..Predel, 1..Predel] of real;
  i,n:integer;
Begin
writeln('введите размерность массива(n>0)и(n<=',Predel,':');
readln(n);
writeln('введите элементы массива:');
For i:=1 to n do
   For j:=1 to n do
       Begin
       write('M[',i,',',j,']=');
       readln(M[i,j])
       end
End.
```

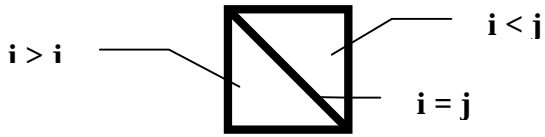
Рисунок 3.5

3.4 Примеры работы с двумерными массивами

Рассмотрим примеры обработки двумерных массивов найти сумму элементов, стоящих на главной диагонали в квадратной матрице

Обращение к элементам главной диагонали

1 –й способ



Рисунок

```
s:=0;
for i := 1 to N do
      s := s + A[i, i];
writeln('сумма = ', s);
```

2 – й способ

```
s:=0;
for i := 1 to N do
   for j := 1 to N do
      if (i=j) then s := s + A[i, j];
writeln('сумма = ', s);
```

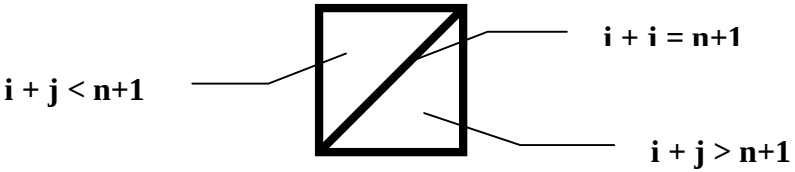
Обращение к элементам выше главной диагонали

```
s:=0;
for i := 1 to N do
   for j := 1 to N do
      if (i < j) then s := s + A[i, j];
writeln('сумма = ', s);
```

Обращение к элементам ниже главной диагонали

```
s:=0;
for i := 1 to N do
   for j := 1 to N do
      if (i > j) then s := s + A[i, j];
writeln('сумма = ', s);
```

Обращение к элементам побочной диагонали



1-й способ

```
s:=0;
for i := 1 to N do
      s := s + A[i, n-i+1];
writeln('сумма = ', s);
```

2-й способ

```
s:=0;
for i := 1 to N do
   for j := 1 to N do
      if (i+j = n+1) then s := s + A[i, j];
writeln('сумма = ', s);
```

Обращение к элементам выше побочной диагонали

```
s:=0;
for i := 1 to N do
   for j := 1 to N do
      if (i+j < n+1) then s := s + A[i, j];
writeln('сумма = ', s);
```

Обращение к элементам ниже побочной диагонали

```
s:=0;
for i := 1 to N do
  for j := 1 to N do
        if (i + j > n+1) then s := s + A[i, j];
writeln('сумма = ', s);
```

3.5 Индивидуальные задания

1. Дана квадратная матрица порядка N. Вычислить среднее арифметическое положительных элементов матрицы, стоящих выше главной диагонали.
2. Дана матрица размерности N на M. Найти в матрице первую по порядку строку с наибольшей суммой элементов. Вывести ее номер.
3. Дана квадратная матрица порядка N. В матрице вычислить среднее арифметическое положительных элементов, стоящих на главной диагонали.
4. Дана квадратная матрица порядка N. Вывести строку матрицы, в которой элемент, стоящий на главной диагонали, максимален.
5. Дана матрица размерности N на M. Положительные элементы матрицы переписать подряд в одномерный массив B.
6. Дана матрица размерности N на M. Вычислить количество строк матрицы, в которых есть хоть один отрицательный элемент.
7. В квадратной матрице найти сумму элементов побочной диагонали и разделить на полученную сумму все элементы последнего столбца.
8. Дана квадратная матрица порядка N. Найти произведение элементов побочной диагонали квадратной матрицы.
9. Дана матрица размерности N на M. Вывести номера всех столбцов матрицы, не содержащих отрицательных элементов.
10. Дана матрица размерности N на M. В матрице найти первый по порядку столбец с максимальной суммой элементов. Вывести его номер.
11. Дана квадратная матрица порядка N. Вывести столбец матрицы, в котором элемент, стоящий на главной диагонали, минимален, среди элементов главной диагонали.
12. Дана матрица размерности N на M. В матрице найти первый по порядку столбец с минимальной суммой модулей его элементов. Вывести его номер.
13. Найти сумму положительных элементов квадратной матрицы, находящихся ниже главной диагонали.

14. Дана квадратная матрица порядка N. Найти максимальный и минимальный элементы матрицы и поменять местами соответствующие им строку и столбец (строка для максимального элемента, столбец для минимального элемента).

15. Дана квадратная матрица порядка N. Найти количество четных элементов квадратной матрицы, расположенных ниже побочной диагонали.

16. Дана матрица размерности N на M. Найти произведение максимальных четных элементов столбцов матрицы.

4 ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ПРОЦЕДУР И ФУНКЦИЙ

4.1 Понятие подпрограммы

Часто в программе обнаруживаются повторяющиеся или похожие фрагменты, которые выполняют одни и те же вычисления, но с различными данными. Такие части программ целесообразно оформлять в виде подпрограмм.

Использование подпрограмм позволяет: улучшить структуру программы, сделать основную программу более наглядной и компактной, уменьшает вероятность ошибок, облегчает отладку.

В языке Паскаль для организации подпрограмм используются процедуры и функции. Процедура - независимая часть программы, предназначенная для выполнения определенных действий. Функция аналогична процедуре, но имеет два отличия: она возвращает в программу некоторый результат и может использоваться как часть выражения. Все процедуры и функции языка Паскаль разделяются на встроенные (стандартные) и пользовательские (создаваемые программистом) процедуры и функции. Встроенные (стандартные) процедуры и функции являются частью языка и могут вызываться по имени без предварительного определения в разделе описаний программы. Процедуры и функции пользователя организовываются самим программистом в соответствии с синтаксисом языка и их предварительное описание (перед использованием) в тексте программы обязательно.

4.2 Использование процедур

Процедуры состоят из группы операторов, реализующих некоторую часть задачи и вызываемых по имени при необходимости в любом месте программы. Перед использованием (вызовом) процедуру необходимо описать согласно следующему формату:

Procedure <имя> **[**формальные параметры**];**
<раздел описаний>
Begin

<основная часть процедуры– раздел операторов >
End;

Описание процедуры включает заголовок - <имя> и тело процедуры - <основная часть процедуры>. Заголовок состоит из зарезервированного слова Procedure, идентификатора (имени) процедуры и необязательного списка формальных параметров с указанием типа каждого параметра. *Например:*

Procedure Korrect; (процедура без формальных параметров)
Procedure Sort **(a:integer);** (a – формальный параметр).

Имя процедуры – идентификатор, уникальный в пределах конкретной программы. Тело процедуры представляет собой блок, по структуре аналогичный блоку обычной программы. Раздел операторов всегда начинается зарезервированным словом Begin, далее следуют операторы, отделенные ";". Завершает раздел зарезервированное слово End и ";". Приведем пример процедуры, которая реализует вывод элементов двумерного массива на экран.

```
Type Mas= array[1..10,1..10] of real;
Procedure Vivod (A: Mas);
       Var i,j:integer;
       Begin {начало процедуры}
        for i:=1 to n  do
           begin
          for j:=1 to  n do
              write(A[i,j]:5);
                writeln;
           end;
       End; {конец процедуры}
...{продолжение основной программы}
```

Рисунок 4.1

Для обращения к процедуре используется оператор вызова процедуры. Он состоит из идентификатора (имени) процедуры и списка фактических параметров, отделенных друг от друга запятыми и заключенных в скобки.

Формат вызова процедур:

<**Имя процедуры**> **(**список фактических параметров**);**

Вызов процедуры, описанной выше, Vivod(B), где B - имя массива типа Mas, который необходимо вывести на экран.

4.3 Использование функций

Формат описания функций:

Function <имя> [формальные параметры]**: <тип результата>;**
<раздел описаний >
Begin
<раздел операторов>;
end;

Функция, определяемая в программе, состоит из заголовка и тела функции. Заголовок содержит зарезервированное слово Function, идентификатор (имя) функции и необязательный список формальных параметров с указанием типа каждого параметра, а также тип возвращаемого функцией значения: тип результата.

Например:

```
Function Prov(x,y,t:integer):integer;
Function Logic:boolean;
```

Имя функции - уникальный в пределах программы идентификатор. Возвращаемый результат может иметь любой скалярный тип. Тело функции представляет собой блок, по структуре аналогичный блоку обычной программы. В разделе операторов должен находиться по крайней мере один оператор, присваивающий функции значение. Если таких присваиваний несколько, то результатом выполнения функции будет значение последнего оператора присваивания указанного в теле функции. Обращение к функции осуществляется по имени с необязательным указанием списка аргументов. Каждый аргумент должен соответствовать формальным параметрам, указанным в заголовке, и иметь тот же тип.

Приведем пример функции, которая возвращает значение максимального элемента в двумерном массиве с размерностью n на m,

```
Type Mas= array[1..10,1..10] of real;
function Max_el (n, m:integer; M: Mas):real;
var i,j:integer; max:real;
begin
     max:=M[1,1];
     for i:=1 to n do
         for j:=1 to m do
             if M[i,j] > max then max:= M[i,j];
Max_el:=max;
end;
```

Рисунок 4.2

4.4 Программирование алгоритмов с использованием подпрограмм

Пример1. Заданы массивы K(2,2), L(3,3) и M(2,2). Вывести на печать номера строк и столбцов, на пересечении которых находятся элементы с наименьшим значением, а также значения этих элементов.

```
Program pr1 ( input. output );
Type mas=array[1..3,1..3] of integer;
Var  K,L,M: mas;
       im,jm: integer; (*номер отроки и столбца*)
       min: integer;  (*минимальное значение элемента*)
       i,j: integer;
  (* подпрограмма *)
procedure minim (a :mas; n: integer; var im,jm ,min: integer);
var i, j: integer;
 begin
  min:=a[1,1 ]; im:=1 ; jm:=1;
  for l:=1 to n do
    for j:=1 to n do
       if a[i,j]<min then
            begin
              min:=a[i,j]; im:=i;jm:=j
            end
end;
(* 0сновная программа *)
begin
     for i:=1 to 2 do
       for j:=1 to 2 do  readln(K[i,j],M[i,i]);
     for i:=1 to 3 do
        for j:=1 to 3 do read(L[i,j]);
      minim (K,2,im,jm,min);
      writeln(' массив K:',im,' ',jm,' ',min);

      minim (L, 3, im,jm,min);
      writeln(' массив L:',im,' ',jm,' ',min):
      minim(M,2,im,jm,min);
      writeln(' массив M:',im,' ',jm,' ',min)
end.
```

Пример 2. Вычисление определённого интеграла методами трапеций и Симпсона.

Задана функция y(x). Требуется найти определённый интеграл с точностью 10^{-3} на заданном в варианте отрезке и заданным методом. Отрезок [a,b]

разбивается на n отрезков. Вычисляется приближённое значение интеграла по следующим формулам:

Для метода Симпсона: $$\int_a^b ydx = \frac{b-a}{3n}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + ... + 2y_{n-2} + 4y_{n-1} + y_n)$$

Для метода трапеций: $$\int_a^b ydx = \frac{b-a}{n}(\frac{1}{2}y_0 + y_1 + y_2 + y_3 + ... + y_{n-1} + \frac{1}{2}y_n)$$

Для метода прямоугольников: $$\int_a^b ydx = \frac{b-a}{n}(y_1 + y_2 + y_3 + ... + y_{n-1} + y_n)$$

Примечание: в методе прямоугольников значения функции вычисляются в середине каждого отрезка разбиения.

Приведем программу решения задачи нахождения интеграла для функции на отрезке методом прямоугольников (рис. 2.1)

```
program e;
 uses crt;
 var i, n, nmax: integer;
     a, b, rez, rez1, eps: real;
 procedure integ(a, b: real; n:integer; var rez:real);
  var i: integer; x,h,S:real;
   begin
     h:=(b-a)/n;
     x:=a+h/2;
     S:=0;
     for i:=1 to n do
      begin
       x:=x+h;
       S:=S+sqrt(2*exp(x)-1)/(x*x+1);
      end;
     rez:=h*S;
   end;
 begin
  writeln(' Введите границы отрезка  ');
  readln(a,b);
  writeln(' Введите начальное число разбиений ');
  readln(n);
  eps:=0.01; nmax:=12000;
  integ(a,b,n,rez);
  repeat
    rez1:=rez; n:=n*2;
    if n > nmax then begin
              writeln('требуемая точность не достигнута');
              halt
            end;
```

```
    integ(a,b,n,rez);
  until abs(rez-rez1)<=eps;
  Writeln('На отрезке [',a:3:1,',',b:3:1,'] интеграл = ',
rez:6:3)
 End.
```

4.5 Практические задания

Задание1. Ввести одномерный массив. Вычислить для каждого элемента массива значение функции по указанной формуле. Найти сумму (произведение) элементов массива или минимальное (максимальное) значение среди исходных данных. Распечатать результаты вычислений.
Задание выбирается из таблицы 1.2 по предпоследней цифре учебного шифра.

Таблица 4.1

	Задание
0	Ввести массив *Z* [1..5] и число *C*. Для каждого элемента массива вычислить $T = \sin^2 Z_i + C$. Найти сумму элементов массива *Z*.
1	Ввести массив *H* [1..5] и число *A*. Для каждого элемента массива вычислить $Z = \frac{\sqrt{H_i}}{A}$. Найти минимальный элемент в массиве *H*.
2	Ввести массив *A* [0..5] и число *P*. Для каждого элемента массива вычислить $B = e^{\frac{Ai}{P}}$. Найти произведение элементов массива *A*.
3	Ввести массив *T* [1..4] и число *g*. Для каждого элемента массива вычислить $f = \sqrt[3]{T_i \cdot g}$.Найти произведение элементов массива *T*.
4	Ввести массив *F* [0..6] и число *C*. Для каждого элемента массива вычислить $G = tg \frac{F_i}{F_i + C}$. Найти сумму элементов массива *F*.
5	Ввести массивы *T*[1..4] и *C*[1..4]. Для каждого *T*[*i*],*C*[*i*] вычислить $d = 3.2 \cdot T_i^3 + C_i$. Найти максимальные элементы в массивах *C*, *T*.
6	Ввести массив *Z* [0..3] и *A* [0..3] . Для каждого *Z*[*i*] и *A*[*i*] вычислить $f = \sqrt{(z_i + a_i)^3}$. Найти сумму значений элементов массивов *Z* и *A*.
	Ввести массив *B* [1..5]. Для каждого элемента массива вычислить

7	$\dot{O} = \sqrt[3]{3 \cdot B_i}$. Найти минимальный элемент в массиве B.
8	Ввести массив L [1..4] и число P. Для каждого элемента массива вычислить $C = cos^2\left(\frac{P}{P - L_i}\right)$. Найти произведение значений элементов массива L.
9	Ввести массив A [1..5]. Для каждого элемента массива вычислить $P = A_i + \sqrt{\frac{A_i}{2}}$. Найти сумму значений элементов массива A.

Задание 2. Найти приближённое значение определённого интеграла с точностью 10^{-3} на заданном в варианте отрезке и заданным методом. Задание выбирается из таблицы 4.2 по последней цифре учебного шифра.

Таблица 4.2

№ п/п	Вычисляемая функция	Нижний и верхний пределы интегрирования	Метод численного интегрирования
0	$y(x)= \frac{x \cdot \ln^2(x+1)}{\sqrt[3]{x^2 + \lg x}}$	a=1.4; b=3.5	прямоугольников
1	$y(x)= \sqrt{e^x - 1}$	a=0.2; b=2.0	Симпсона
2	$y(x)= x^3\sqrt{1+x}$	a=1.0; b=5.0	трапеций
3	$y(x)= e^x \cdot \sin(x)$	a=0; b=π/2	прямоугольников
4	$y(x)= \sqrt{3^x - 1}$	a=0.2; b=4.0	Симпсона
5	$y(x)= 1/(1+x^2)$	a=1; b=3	трапеций
6	$y(x)= \sqrt{1 - \frac{1}{4} \cdot \sin^2 x}$	a=0 ; b= π/2	Симпсона
7	$y(x)= \sqrt{6x - 5}$	a=1; b=9	трапеций
8	y(x)=x sin x	a=0; b=2π	Симпсона
9	$y(x)= \lg(x^2+3,5)$	a=2,1; b=6,3	трапеций

4.6 Контрольные вопросы

1. Для чего предназначены подпрограммы?
2. Какие встроенные процедуры и функции Вы использовали?.
3. Что включает в себя заголовок процедуры?
4. Что такое локальные и глобальные переменные?

5. Формальные и фактические параметры.
6. Параметры-значения и параметры-переменные
7. Чем отличается описание функции от описания процедуры?
8. Чем отличается обращение к функции от обращения к процедуре?

ЛИТЕРАТУРА

1. *Абрамов С. А., Зима В. С.* Начало программирования на языке ПАСКАЛЬ. – М.: Наука, 1987. – 112 с.
2. *Вальвачев А. Н., Крисевич В. С.* Программирование на языке Паскаль для персональных ЭВМ ЕС: Справочное пособие. – Мн.: Вышэйшая школа, 1989. – 223 с.
3. *А.П. Кейзер, Т.Е. Кабакова, З.Н.Рогачева, С.Г. Халамов* Решение задач контрольной работы средствами математического пакета MathCad и табличного процессора Excel. – Гомель, 2001. – 14 с.
4. Информатика. Программирование на языке Паскаль: Практикум по лабораторным работам. Ч.1 /*.А.П. Кейзер, Ю.А. Пшеничнов, М.В. Борисенко, О.И. Еськова;* Под ред. Ю.А. Пшеничнова – Гомель:БелГУТ, 2001.- 46с.
5. *Фаронов В.В.* Программирование на персональных ЭВМ в среде Турбо-Паскаль. – М.: Изд-во МГТУ, 1990. – 590 с.
6. *Фаронов В.В. Турбо Паскаль 7.0.* Начальный курс: Учеб. пособие. – М.: "Нолидж", 1998. – 616 с.