МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Белорусский национальный технический университет

Кафедра «Металлургия литейных сплавов»

ИНФОРМАТИКА

Лабораторный практикум
для студентов специальности
1-42 01 01 «Металлургическое производство
и материалообработка»

В 2 частях

Часть 2

Минск
БНТУ
2013

УДК 004(075.8)
ББК 32.97.7
И74

Составители:
И. В. Рафальский, А. В. Арабей

Рецензенты:
М. М. Татур, В. Д. Василенок

И74 **Информатика** : лабораторный практикум для студентов специальности 1-42 01 01 «Металлургическое производство и материалообработка» : в 2 ч. / сост. : И. В. Рафальский, А. В. Арабей. – Минск : БНТУ, 2009–2013. – Ч. 2. – 2013. – 69 с.
ISBN 978-985-550-251-8 (Ч. 2).

Лабораторный практикум предназначен для закрепления и углубления теоретической базы знаний студентов при изучении дисциплин «Информатика», «Прикладная информатика», а также приобретения практических навыков работы с современными электронно-вычислительными и программными средствами, алгоритмами и компьютерными методами поиска, использования, хранения и обработки информации.
Часть 1 вышла в БНТУ в 2009 году.

УДК 004(075.8)
ББК 32.97.7

ISBN 978-985-550-251-8 (Ч. 2)
ISBN 978-985-525-213-0

Лабораторная работа № 1

МАТРИЧНОЕ ИСЧИСЛЕНИЕ: ДЕЙСТВИЯ НАД МАТРИЦАМИ

Матрица – математический объект, записываемый в виде прямоугольной таблицы элементов (например, целых, действительных или комплексных чисел), которая представляет собой совокупность строк и столбцов, на пересечении которых находятся ее элементы. Количество строк и столбцов матрицы задают размер матрицы.

Для матрицы определены следующие *алгебраические операции*:

– сложение матриц, имеющих один и тот же размер;

– умножение матриц соответствующего размера (матрицу, имеющую n столбцов, можно умножить справа на матрицу, имеющую n строк);

– умножение на матрицу вектора (по обычному правилу матричного умножения; вектор является частным случаем матрицы);

– умножение матрицы на число.

Также выполняют *элементарные преобразования матрицы*:

– умножение какой-нибудь строки (столбца) на отличное от нуля число;

– прибавление к какой-нибудь строке (столбцу) другой ее строки (столбца), умноженной на произвольное число;

– перестановку местами любых двух строк (столбцов).

Сложение матриц. Суммой матриц $A = (a_{ij})$ и $B = (b_{ij})$ одинаковых размеров называется матрица $C = (C_{ij})$ тех же размеров, у которой $C_{ij} = a_{ij} + b_{ij}$. Обозначение: $C = A + B$.

Вычитание матриц. $A - B = A + (-B)$.

Умножение матрицы на число. Произведением матрицы $A = (a_{ij})$ на число b называется матрица $C = (C_{ij})$ тех же размеров, у которой $c_{ij} = ba_{ij}$. Обозначение: $C = bA$.

Умножение матриц. Произведением матрицы $A = (a_{ik})$ размером $m \times n$ на матрицу $B = (b_{ik})$ размером $n \times p$ называется матрица $C = (C_{ij})$ размером $m \times p$, у которой

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + ... + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj} .$$

Обозначение: $C = AB$. Для матриц A и B, где $AB \neq BA$.

Транспонирование матиц. Если

$$A = \begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{21} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{mn} \end{bmatrix},$$

то транспонированная матрица

$$A^r = \begin{bmatrix} a_{11} & a_{21} & ... & a_{m1} \\ a_{12} & a_{22} & ... & a_{m2} \\ ... & ... & ... & ... \\ a_{1n} & a_{2n} & ... & a_{mn} \end{bmatrix}.$$

В программировании матрица представляет собой двумерный массив.

1.1. Операции над векторами

Цель работы: изучить способы описания одномерных массивов (векторов) и приемы работы с ними.

Матрица, состоящая из одной строки или одного столбца, называется соответственно вектор-строкой или вектор-столбцом. Вектор-столбцы и вектор-строки называют просто *векторами*:

$[a_{11} \quad a_{12} \quad ... \quad a_{1n}]$ – матрица-строка;

$$\begin{bmatrix} a_{11} \\ a_{21} \\ \dots \\ a_{m1} \end{bmatrix}$$ – матрица-столбец.

В программировании вектор – это одномерный массив. Чтобы использовать массив в программе на языке Паскаль, требуется определить параметры массива и описать переменную-массив. Описание типа массива задается следующим образом:

<*имя массива*> = array [*тип индексов*] of [*тип элементов массива*];

Тип индексов массива заключается в квадратные скобки после служебного слова array, а тип элементов (компонентов) массива задается после служебного слова of. Для типа индексов обычно используют тип-диапазон, который определяет границы изменения индексов.

Ниже приведены примеры описания одномерных массивов:

```
var
      f :array[1..20] of integer;
      mas:array[0..50] of word;;
      a:array[-7..7] of real;
      dis:array[1..30] of byte;
```

Как и тип-диапазон, перечисляемый и структурированные типы можно, а иногда необходимо объявлять в разделе описания типов (например, при описании формальных параметров в подпрограммах), который начинается с ключевого слова *type*:

```
type
   mas1 = array[1..20] of real;
   mas2 = array[0..40] of byte;
  var
   a:mas1;
   b,c:mas2;
```

В разделе описания типов с типом сопоставляется некоторое имя и в дальнейшем вместо явного указания типа можно использовать введенный для данного типа идентификатор. Имя типа обозначается идентификатором, а сам тип описывается согласно определенным для него правилам.

Чаще всего в реальных задачах индексы изменяются от 1. В этом случае индекс элемента совпадает с его порядковым номером. Размерность массива – величина произвольная, однако суммарная длина любого массива не может быть больше 65520 байт. Если количество элементов массива неизвестно, то этой величиной можно задаться через раздел описания констант:

```
Const   N = 5;
Var   x, y : array [1..N*10] of real;
      z : array [1..N, 1..N*2] of integer;
```

Обращение к элементам массива осуществляется по их индексам, т.е. при упоминании в программе любого элемента массива сразу за именем массива должен следовать индекс элемента в квадратных скобках, например:

```
f[1]:=0;
mas[100]:=555.
```

Ввод/вывод одномерных массивов

Определение значений элементов массивов может быть произведено различными способами: с использованием клавиатуры, файлов, оператора присваивания.

Способ задания элементов массива с использованием оператора присваивания:

```
Begin
B[1]:=1.6;
B[2]:=18.1;
B[3]:=0.04;
```

или

```
Const
   A : array [1..3] of real = ( 1.6, 18.1,0.04);
```

Ввод массива поэлементно с клавиатуры осуществляется с помощью оператора ввода *read*:

```
begin
  read(B[1]);
  read(B[2]); ...
```

Недостаток описанных способов – громоздкость программного кода.

Так как операции ввода/вывода относятся к алгоритмам последовательной обработки элементов массива, чаще всего используется организация ввода с помощью операторов цикла:

```
for i:=1 to 5 do
begin
write('введите A(' , i ')=');
readln( A[i] );
end;
```

При использовании в программе такого способа ввода элементов массива будет необходимо пять раз, по мере остановки программы и запроса вводимого элемента, набирать значения элементов массива и нажимать клавишу Enter.

Примеры обработки одномерных массивов

При обработке одномерных массивов чаще всего возникают такие задачи, как нахождение суммы элементов массива, произведения, среднего значения, поиск заданного элемента массива, сортировка элементов массива, ввод и вывод элементов массива и т.д.

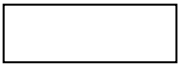
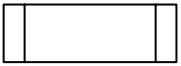
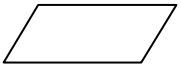
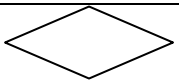
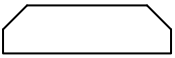
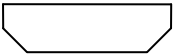
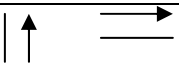
Пример 1. формирования элементов массива с помощью датчика случайных чисел, вычисление максимального значения и суммы всех элементов массива, вывод элементов массива и результатов вычислений на экран, приведен ниже.

```
program primer1;
  const k=100;                    {максимальный размер массива}
 type       mas = array[1..k] of integer;
    var   a:mas;   n,s:integer;
          i: integer;             {индекс элемента массива}
         max:integer;             {индекс максимального элемента}
     begin
        writeln('Введите число элементов массива:');
        read(n);
   randomize;                     {инициализация датчика случайных чисел}
      for i:=1 to n do
      A[i]:=random(10);           {формирование массива с помощью
                                   датчика случайных чисел}
writeln(' массив A[i]');
      for i:=1 to n do
       write (A[i]:5);
       writeln;
       s:=0;                      {нахождение суммы }
       for i:=1 to n do
          s:=s+A[i];
      writeln('s=',s);
    max:=A[1];                    {предполагаем, что первый
                                   элемент максимальный}
      for i:=1 to n do
if A[i]>max then max:= A[i];
  writeln('максимальный элемент массива =',max);
    end.
```

Логику вычислительного процесса удобно представлять в виде алгоритмов. Наиболее наглядным является графический способ (в виде блок-схемы) описания алгоритмов, когда каждое действие в программе описывается условным графическим обозначением (символом). В блок-схеме символы располагаются в определенной последовательности и соединяются линиями связи. Внутри символов в произвольной форме делаются поясняющие записи. Основным направлением линий связи являются сверху вниз и слева направо. Если направление линии совпадает с основным направлением блок-схемы, то стрелка на ней не проставляется. Элементы блок-схемы могут иметь нумерацию, которая расставляется в соответствии с основными направлениями линий. Линии связи элементов

блок-схемы, как правило, не должны иметь пересечений. Если необходимо пересечь линии связи элементов блок-схемы, их разрывают, а в концах разрыва проставляют соединители с указанием номера символа, к которому направлена данная линия. При больших размерах блок-схемы ее располагают на нескольких листах. В этом случае на концах разрыва линий связи проставляют межстраничные соединители.

Символы блок-схем (алгоритмов)

Логическое начало или логический конец алгоритма	
Вычислительный процесс	
Подпрограмма	
Ввод-вывод данных	
Выбор, развилка, условие	
Начало цикла	
Конец цикла	
Соединители	
Межстраничный соединитель	

Алгоритм, содержащий хотя бы одно условие, называется разветвляющимся вычислительным процессом. В зависимости от условия будет выполняться та или другая последовательность действий, называемых ветвями алгоритма. Количество ветвей и условий в алгоритмах не ограничено.

На рисунке 1 представлена блок-схема (алгоритм) описанной в примере задачи.

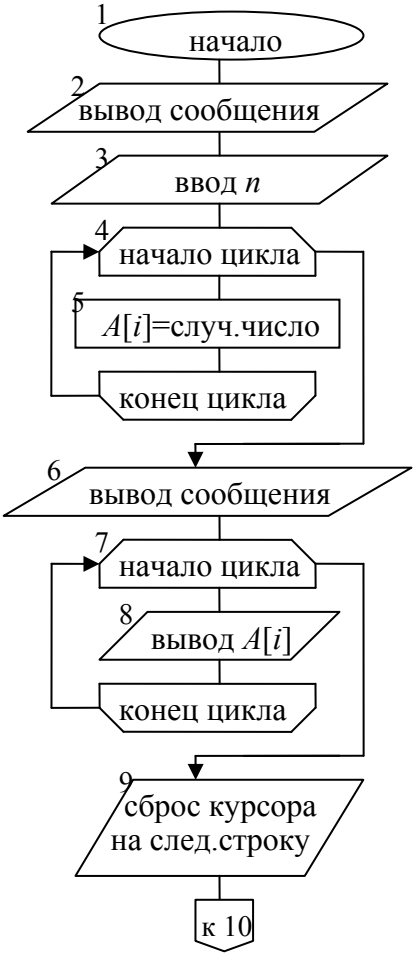


Рисунок 1 – Блок-схема (алгоритм) задачи

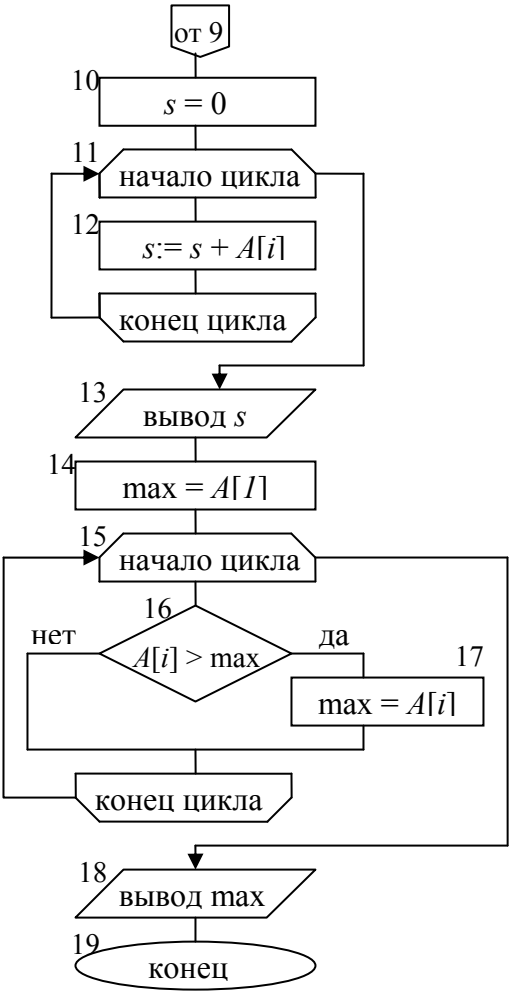


Рисунок 1 (окончание)

Пример 2. Сортировка элементов одномерного массива в порядке возрастания. Суть сортировки состоит в том, что сравниваются соседние элементы и находится минимальный элемент массива, который определяется первым элементом массива. Далее находят следующий минимальный элемент, так же его выставляют вслед за первым элементом и так далее.

```
program primer2;
 const k=100;                  {максимальный размер массива}
 type
   mas = array[1..k] of real;
 var
    C:mas;
    i,j:integer;               {индекс элемента массива}
    n:integer;                 {число элементов массива}
    x:real;                    {дополнительная переменная для
                                перестановки местами двух элементов
                                массива}
  begin
     writeln('Введите число элементов массива:');
     read(n);
                               {ввод массива}
     for i:=1 to n do
      begin
       write ('Введите ',i,'-й элемент массива:');
       read(C[i])
      end;
     writeln('исходный массив');
     for i:=1 to n do
      write (C[i]:5);
     writeln;
                               {сортировка элементов массива }
     for i:=1 to n-1 do
     for j:=i+1 to n do
      if C[i]>C[j] then        {сравниваем два соседних элемента}
          begin                {меняем местами соседние элементы}
           x:=C[i];
           C[i]:=C[j];
           C[j]:=x;
          end;
     writeln(' массив после сортировки:');
     for i:=1 to n do
      write (C[i]:5:2);
     writeln;
end.
```

Задания для самостоятельной работы

1. Составить алгоритм и программу вывода на экран максимального элемента одномерного массива размерностью N и номера, под которым он находится. Значение N не должно превышать 30.

2. Составить алгоритм и программу сортировки по возрастанию и убыванию одномерного массива данных, состоящего из 50 чисел, выбранных произвольно. Организовать вывод результата на экран.

3. Составить алгоритм и программу вывода на экран максимального и минимального значений одномерного массива, состоящий из 10 элементов, выбранных произвольным образом. Поменять местами значения 1-го и 5-го элементов и вывести результат на экран.

4. Дан одномерный массив, состоящий из 30 элементов, выбранных произвольным образом. Определить, есть ли в массиве два одинаковых элемента и вывести их номера на экран.

5. Составить алгоритм и программу вывода на экран одномерного массива данных до и после преобразования, состоящий из 25 элементов, выбранных произвольным образом. Заменить все четные элементы их квадратами, а нечетные удвоить.

6. Составить алгоритм и программу вывода на экран одномерного массива, состоящий из 20 элементов, выбранных произвольным образом из интервала (0; 55). Найти сумму элементов массива, принадлежащих промежутку от 20 до 30. Вывести результат на экран.

7. Составить алгоритм и программу вывода на экран суммы нечетных элементов одномерного массива данных, состоящего из N элементов, выбранных произвольным образом.

8. Составить алгоритм и программу поиска среднего арифметическое значения элементов массива, состоящего из N чисел. Элементы массива вводятся с клавиатуры.

1.2. Операции над матрицами

Цель работы: изучить способы описания двумерных массивов данных (матриц) и приемы работы с ними.

Двумерные массивы имеют аналогию с таким понятием в математике как матрица. В языке Паскаль двумерный массив – это массив, элементами которого являются одномерные массивы:

```
C: array[1..n] of array[1..m] of real;
```

или

```
C: array[1..n,1..m] of real;
```

Как правило, пользуются вторым способом описания массива, где *n* – количество строк, *m* – количество столбцов матрицы. Чаще в качестве идентификатора номера строки используют символ *i*, а столбца – *j*. Тогда к элементу массива, описанному выше, можно обратиться по имени *C[i,j]*.

Примеры описания двумерных массивов и обращения к их элементам:

```
type
  mas1=array[1..n] of real;
  mas2=array[1..7,1..9] of integer;
var
  x:array[1..5] of mas1;
  mass:array[0..20,1..40] of byte;
  c:mas2;
  asd:array[1..20,1..10] of integer;
begin
  x[1,1]:=12.1;
  mass[0,1]:=5;
  c[4,6]:=34;
  asd[1,8]:=0;
```

При обработке матриц следует учитывать особенности работы вложенных циклов. Она отличается тем, что на каждый шаг внешнего цикла внутренний цикл производит полное число шагов. Например:

```
Var   i,j: byte;
            c:array[1..3,1..5]of real;
begin
 for i:=1 to 3 do
          begin
            for j:=1 to 5 do
               begin
```

```
        c [i,j] := cos( X[i,j] );
      end
end;
……
```

После того, как внешний цикл сделает первый шаг (переменная *i* примет значение *1*), во внутреннем цикле переменная *j* пробежит все положенные ей значения от *1* до *5*. Когда внутренний цикл завершится, во внешнем цикле переменная *i* изменится на шаг, т.е. примет значение *2*. После чего внутренний цикл начнет работать сначала, т.е. *j* изменится от *1* до *5* и т.д.

Ввод/вывод двумерных массивов

Ввод данных в простейших случаях может осуществляться теми же способами, как и в случае одномерных массивов данных:

– присваиванием:

```
Begin
   C[1,1]:=5.7; C[1,2]:=2.5; ....
```

– через раздел описания констант (значения элементов по каждому отдельному измерению (строки или столбцы) отделяются друг от друга дополнительными скобками:

```
Const
C: array [1..2,1..2] of real =  ( ( 1.3, 3.7 ) , ( 4.0, 1.1 ));
```

В приведенном примере значения элементов матрицы располагаются в следующем порядке: ((C[1,1], C[1,2]), (C[2,1], C[2,2])), т.е. по строкам;

– с помощью операторов ввода и циклов:

```
for i:=1 to 3 do
      for j:=1 to 5 do
begin
write('введите C[' , i, ' ' ,j, ']=');
readln( C[i,j] );
end;
```

Примеры обработки двумерных массивов

При обработке двумерных массивов возникают такие же задачи, как и при обработке одномерных массивов: нахождение суммы элементов массива, произведения, среднего значения, поиск некоторого элемента в массиве, ввод и вывод элементов массива и т.д.

Пример 1. Формирование элементов массива с помощью датчика случайных чисел, вывод массива на экран, вычисление суммы всех элементов матрицы.

```
program primer3;
const n1=50; n2=50;
type   mas = array[1..n1,1..n2] of integer;
 var C:mas;
      i, j, n, sum, m : integer;
  begin
     writeln('Введите число строк и столбцов массива:');
     read(n,m);
     randomize;
for i:=1 to n do
    for j:=1 to m do
        C[i,j]:=random(20);
writeln('полученный массив');
     for i:=1 to n do
      begin
       for j:=1 to m do
         write (C[i,j]:5);
       writeln;
      end;
     sum:=0;
     for i:=1 to n do
     for j:=1 to m do
        sum:=sum+C[i,j];
     writeln('sum=',sum);
  end.
```

Пример 2. Выполнить действия над матрицами A и B: $A \times B + 2B$, где

$$A = \begin{bmatrix} 0.98 & 3.71 & 1.97 & 12.01 \\ 0.18 & 7.80 & 14.56 & 14.19 \\ 0.53 & 0.76 & 18.53 & 12.82 \\ 0.17 & 0.09 & 7.37 & 48.77 \end{bmatrix}, \qquad B = \begin{bmatrix} 3.24 \\ 5.40 \\ 43.67 \\ 64.09 \end{bmatrix}.$$

Задачу необходимо решить:

2.1. с использованием пакета MS Excel;

2.2. с использованием средств программирования Паскаль.

Задание 2.1. Введем исходные данные на рабочий лист MS Excel (рисунок 2).

F16 | {=МУМНОЖ(B1:E4;G1:G4)+2*G1:G4}

	A	B	C	D	E	F	G
1		0,98	3,71	1,97	12,01		3,24
2	A=	0,18	7,8	14,56	14,19	B=	5,4
3		0,53	0,76	18,53	12,82		43,67
4		0,17	0,09	7,37	48,77		64,09
9		878,96					6,48
10	AB=	1587,976				2B=	10,8
11		1636,66					87,34
12		3448,554					128,18
16			885,44				885,44
17		AB+2B=	1598,776			AB+2B=	1598,776
18			1724				1724
19			3576,734				3576,734

Рисунок 2 – Рабочее окно MS Excel: действия над матрицами

Для умножения матрицы *A* на матрицу *B*, выделим диапазон **B9:B12** и воспользуемся функцией **МУМНОЖ(B1:E4;G1:G4).**

Умножение (деление) матрицы на число можно выполнить при помощи элементарных операций. В нашем случае необходимо умножить матрицу из диапазона **G1:G4** на число 2. Выделим ячейки

G9:G12 и введем формулу **=2*G1:G4.** Сложение матриц выполняется аналогично. Выделим диапазон **D16:D19** и введем формул **=B9:B12+G9:G12.**

Для получения результата в обоих случаях необходимо нажать комбинацию клавиш **Ctrl+Shift+Enter.** Кроме того, в строке формул рабочего листа, изображенного на рисунке 2, показано как можно вычислить матрицу *C* одним выражением.

Задание 2.2.

```
program matrica;
uses crt;
        const n=1;  m=4;
        a: array[1..m,1..m] of real = (
        ( 0.98, 3.71, 1.97, 12.01),
        ( 0.18, 7.80, 14.56, 14.19),
        ( 0.53, 0.76, 18.53, 12.82),
        (0.17, 0.09, 7.37,  48.77));
        b:  array[1..m,1..n] of real = ((3.24),(5.40), (43.67), (64.09));
        var i,j,k:integer;
        c:  array[1..m,1..n]  of real;
        Begin
        clrscr;
        {Вычисление значений элементов матрицы C}
         for i:=1 to m do
        for j:=1 to n do begin
        c[i,j ]:=0;
        for k:=1 to m do
        c[i, j] :=c[i, j]+a[i,k]*b[k, j];
        c[i,j]:=c[i,j]+2*b[i,j];
        end;
        {Вывод на экран матрицы C}
        writeln('Матрица C:');
        for i:=1 to m do begin
         for j:=1 to n do
        write(c[i,j] : 7:2);
        writeln;          end;
      End.
```

Задания для самостоятельной работы

1. Составить алгоритм и программу вывода на экран матрицы 10×10. Элементы матрицы на главной диагонали равны единице, остальные – нулю.

2. Составить алгоритм и программу поиска одномерного массива B, состоящего из суммы элементов каждой строки матрицы $A(7; 9)$. Найти максимальный элемент матрицы A в 6-й строке. Заменить последний элемент в 4-й строке матрицы A на найденный максимальный элемент. Организовать вывод результата на экран.

3. Составить алгоритм и программу поиска наименьшего элемента двумерного массива и номер строки, в которой он находится. Массив состоит из целых чисел, элементы которого вводятся с клавиатуры. Размер массива $m \times n$ представлен в таблице 1.

Таблица 1

№ варианта	1	2	3	4	5	6	7	8	9	10
m	2	6	3	5	7	3	9	3	7	9
n	6	5	10	5	5	8	3	6	4	8

4. Составить алгоритм и программу поиска суммы элементов главной диагонали матрицы $A(m \times n)$, элементы которой задаются датчиком случайных чисел на интервале [–25; 39]. Размер массива определяется согласно данным, представленным в таблице 1.

5. Составить алгоритм и программу поиска номера столбца массива размером $n \times m$, в котором находится наибольшее количество элементов, кратных 2. Элементы задаются датчиком случайных чисел на интервале [–20; 30]. Размер массива определяется согласно данным, представленным в таблице 1.

6. Составить алгоритм и программу вывода на экран матрицу размером A $(n \times m)$, в которой необходимо поменять местами строку, содержащую элемент с наибольшим значением, со строкой, содержащей элемент с наименьшим значением.

7. Составить алгоритм и программу вывода на экран матрицы A $(n \times m)$, разницы между максимальным и минимальным элеменетом этой матрицы. Принять, что $a_{ij} = i + j^3$; $i = 1, 2, \ldots m$; $j = 1, 2, \ldots n$. Данные для решения задачи представлены в таблице 1.

8. Составить алгоритм и программу поиска среднего арифметического, максимального и минимального значений элементов матрицы $A(m \times n)$. Элементы задаются датчиком случайных чисел на интервале [20; 99]. Размер массива определяется согласно данным, представленным в таблице 1.

9. Составить алгоритм и программу вычисления транспонированной матрицы A^T $(n \times n)$. Элементами матрицы $A(n \times n)$ являются случайные числа.

10. Составить алгоритм и программу расчета элементов матрицы C: $C = A(m \times n) + B(m \times n)$, где a_{ij}, b_{ij} – элементы матрицы A и B, такие что $a_{ij} = i + j^2$; $b_{ij} = (i - j)^3$; $i = 1, 2, \ldots m$; $j = 1, 2, \ldots n$. Организовать вывод на экран максимального и минимального значений элементов матрицы C. Данные для решения задачи представлены в таблице 1.

11. Составить алгоритм и программу расчета элементов матрицы C: $C = A(m \times n) \cdot B(n \times m)$, где a_{ij}, b_{ij} – элементы матрицы A и B, такие что $a_{ij} = i - j^3$; $b_{ij} = (i^2 + j)^5$; $i = 1, 2, \ldots m$; $j = 1, 2, \ldots n$. Организовать вывод на экран максимального и минимального значений элементов матрицы C. Данные для решения задачи представлены в таблице 1.

Лабораторная работа № 2

РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ СРЕДСТВАМИ МАТРИЧНОГО ИСЧИСЛЕНИЯ

Цель работы: освоение практических навыков решения систем линейных уравнений средствами матричного исчисления.

Рассмотрим систему *n* линейных уравнений с *n* неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases};$$

матрица *A*, составленная из коэффициентов при неизвестных:

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix};$$

матрица *B*, составленная из свободных членов:

$$B = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix};$$

матрицу *X* называют матрицей-столбцом из неизвестных:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}.$$

Тогда всю систему линейных уравнений можно записать так:

$$AX = B,$$

где A имеет смысл таблицы коэффициентов a_{ij} системы линейных уравнений.

Матрица A называется основной матрицей. Матрица A со столбцом B – расширенной.

Если матрица A системы невырожденная, т.е. существует обратная матрица A^{-1}, т.е. решение системы линейных уравнений можно найти по формуле $X = A^{-1}B$.

Справочно. Единичная матрица:

$$E = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Матрица A^{-1} – обратная для матрицы A, если $AA^{-1} = A^{-1}A = E$.

Для квадратной матрицы A обратная существует тогда и только тогда, когда ее оределитель не равен 0 ($\det A \neq 0$).

$$A^{-1} = \frac{1}{\det A} \begin{bmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \dots & \dots & \dots & \dots \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{bmatrix},$$

где A_{ij} – алгебраические дополнения элементов a_{ij} матрицы A.

Пример. Необходимо найти решение неоднородной системы линейных уравнений методом определения обратной матрицы с использованием пакета MS Excel:

$$\begin{cases} 0.98x_1 + 3.71x_2 + 1.97x_3 + 12.01x_4 = 3.24 \\ 0.18x_1 + 7.80x_2 + 14.56x_3 + 14.19x_4 = 5.40 \\ 0.53x_1 + 0.76x_2 + 18.53x_3 + 12.82x_4 = 43.67 \\ 0.17x_1 + 0.09x_2 + 7.37x_3 + 48.77x_4 = 64.09 \end{cases}.$$

Эту систему можно представить в матричном виде:

$$AX = B,$$

где $A = \begin{bmatrix} 0.98 & 3.71 & 1.97 & 12.01 \\ 0.18 & 7.80 & 14.56 & 14.19 \\ 0.53 & 0.76 & 18.53 & 12.82 \\ 0.17 & 0.09 & 7.37 & 48.77 \end{bmatrix}$ – матрица коэффициентов системы уравнений;

$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$ – вектор неизвестных уравнений;

$\mathrm{B} = \begin{bmatrix} 3.24 \\ 5.40 \\ 43.67 \\ 64.09 \end{bmatrix}$ – вектор правых частей уравнений.

Введем матрицу *A* и вектор *B* в рабочий лист MS Excel (рисунок 3).

Матрица *A* находится в ячейках **B1:E4,** а вектор *B* в диапазоне **G1:G4.** Для решения системы методом обратной матрицы необходимо вычислить матрицу, обратную к *A***.** Для этого выделим ячейки для хранения обратной матрицы **(B6:E9)** и введем функцию **МОБР(B1:E4),** а затем одновременно нажмем клавиши **Ctrl+Shift+Enter.** Рабочая книга MS Excel примет вид изображенный на рисунке 4.

	A	B	C	D	E	F	G
1	A=	0,98	3,71	1,97	12,01	b=	3,24
2		0,18	7,8	14,56	14,19		5,4
3		0,53	0,76	18,53	12,82		43,67
4		0,17	0,09	7,37	48,77		64,09

Рисунок 3 – Матрица *A* и вектор *B* в рабочем листе MS Excel

	A	B	C	D	E	F	G
1	A=	0,98	3,71	1,97	12,01	b=	3,24
2		0,18	7,8	14,56	14,19		5,4
3		0,53	0,76	18,53	12,82		43,67
4		0,17	0,09	7,37	48,77		64,09
5							
6	A^-1=	0,950443	-0,48437	0,353553	-0,18606		
7		0,030608	0,122851	-0,09221	-0,01904		
8		-0,02916	0,008715	0,054019	-0,00956		
9		0,001037	0,000145	-0,00923	0,022632		

Рисунок 4 – Рабочее окно MS Excel: получение обратной матрицы

Теперь необходимо умножить полученную обратную матрицу на вектор *B*. Выделим ячейки для хранения результирующего вектора, например **H6:H9,** и введем функцию **МУМНОЖ(B6:E9;C1:C4),** а затем одновременно нажмем клавиши **Ctrl + Shift + Enter.**

Для того чтобы проверить, правильно ли решена система уравнений, необходимо умножить матрицу *A* на вектор *x* и получить в результате вектор *B***.** Умножение матрицы *A* на вектор *x* (рисунок 5) осуществляется при помощи функции **МУМНОЖ(B1:E4;H6:H9).**

	A	B	C	D	E	F	G	H	I	J
1		0,98	3,71	1,97	12,01		3,24			3,24
2	A=	0,18	7,8	14,56	14,19	b=	5,4		Проверка	5,4
3		0,53	0,76	18,53	12,82		43,67			43,67
4		0,17	0,09	7,37	48,77		64,09			64,09
5										
6		0,950443	-0,48437	0,353553	-0,18606			3,978917		
7	A^-1=	0,030608	0,122851	-0,09221	-0,01904		x=	-4,48473		
8		-0,02916	0,008715	0,054019	-0,00956			1,699193		
9		0,001037	0,000145	-0,00923	0,022632			1,051756		

Рисунок 5 – Рабочее окно MS Excel:
проверка правильности решения системы уравнений

Задание для самостоятельной работы

Найти решение неоднородной системы линейных уравнений методом определения обратной матрицы с использованием встроенных функций пакета MS Excel и данных, определяемых вариантом:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \qquad\qquad \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}.$$

Значения элементов матриц A и B в соответствии с заданным вариантом:

№ 1

A				B
0.81	7.28	18.05	46.95	91.41
0.48	6.95	12.65	33.43	83.58
0.23	6.69	5.31	46.66	81.30
0.46	9.71	16.01	7.15	51.48

№ 6

A				B
0.12	8.84	4.08	25.39	41.17
0.92	8.94	18.14	10.81	42.02
0.40	2.01	19.54	8.83	14.89
0.68	1.73	9.31	13.02	99.42

№ 2

A				B
0.76	8.13	18.84	12.91	38.06
0.57	2.27	17.95	21.78	35.92
0.43	2.82	7.75	1.33	82.97
0.09	4.27	5.49	36.13	32.68

№ 3

A				B
0.39	9.44	19.53	0.46	95.49
0.41	4.95	16.39	35.89	29.20
0.05	1.61	8.47	27.73	1.75
0.16	6.90	7.51	22.49	66.16

№ 4

A				B
0.73	9.75	11.34	4.01	10.59
0.73	8.72	11.64	35.30	91.69
0.26	5.53	9.71	27.47	97.29
0.81	7.64	11.75	47.85	24.17

№ 5

A				B
0.55	9.40	7.26	5.73	61.13
0.01	7.77	11.83	27.66	96.75
0.31	0.86	2.56	14.58	6.18
0.66	5.69	14.53	8.38	5.15

№ 7

A				B
0.08	2.44	5.10	28.91	21.16
0.53	6.76	3.41	13.17	19.45
0.25	8.94	11.25	30.65	33.63
0.21	0.39	3.28	26.87	52.49

№ 8

A				B
0.18	4.65	5.12	8.76	57.30
0.85	1.13	5.89	30.98	27.17
0.55	7.66	1.97	45.95	68.87
0.73	2.47	2.82	26.84	1.10

№ 9

A				B
0.68	2.93	18.15	11.21	96.89
0.81	8.12	0.94	33.96	78.81
0.77	2.11	8.11	29.90	67.05
0.84	9.20	1.59	37.86	54.94

№ 10

A				B
0.49	1.38	10.53	13.43	26.51
0.82	2.93	3.29	41.30	62.38
0.51	4.87	8.81	6.29	4.67
0.91	4.56	2.65	5.44	42.36

Лабораторная работа № 3

РАБОТА С ФАЙЛАМИ В ПАСКАЛЕ

Цель работы: изучить стандартные подпрограммы языка Паскаль для работы с файлами.

Файл – совокупность данных, записанных во внешней памяти под определенным именем.

Целесообразность применения файлов диктуется следующими причинами:

– размещение большого объема информации, которая может быть подготовлена заранее и применяться неоднократно;

– файл данных может быть подготовлен другой программой;

– программа, использующая данные из файла, не требует присутствия пользователя в момент фактического исполнения.

Для работы с файлами в программе необходимо определить файловую переменную, которую можно описать следующим образом:

```
type
  < имя > = file of < тип >;
  < имя > = text;
  < имя > = file;
```

где <*имя*> – имя файлового типа или файловой переменной;

file, of, text – ключевые слова (в переводе с англ.: файл, из, текст);

< *тип* > – любой тип языка Паскаль, кроме файлового.

В зависимости от способа описания можно выделить текстовые *(text)* файлы, двоичные или типизированные (*file of*) и нетипизированные (*file*). Вид файла определяет способ хранения информации в файле.

Текстовые файлы представляют собой совокупность строк переменной длины с последовательным доступом к данным. Информация в текстовых файлах хранится в символьном виде. Логически последовательный файл можно представить как именованную цепочку байтов, имеющую начало и конец. Последовательный файл отличается от файлов с другой организацией тем, что чтение/запись из файла или в файл ведутся байт за байтом от начала к концу. Рассмотрим основные процедуры и функции для работы с текстовыми файлами.

Процедуры

assign (F, name) – связь файловой переменной *F* с внешним файлом с именем *name* типа string.

reset (F) – открытие для чтения существующего файла с именем, определенным процедурой *assign*. При выполнении этой процедуры файл подготавливается к чтению: внутренняя (указатель файла) устанавливается на начало файла (на его первую компоненту).

rewrite(F) – открытие нового пустого файла для записи с именем, определенным процедурой *assign*. Если в файле уже была информации, процедура удаляет ее (очищает файл) и устанавливает указатель файла на первую компоненту.

read(F,<список ввода>) – чтение информации из файла *F*.

readln(F,<список ввода>) – чтение информации из файла с указанием признака конца строки.

write(F,< список ввода>) – запись информации в файл *F*.

writeln(F,< список ввода >) – запись информации в файл с переходом на новую строку.

close(F) – закрытие открытого файла.

erase(F) – уничтожение внешнего файла, с которым была связана файловая переменная *F*.

mkdir(Path) – создание каталога, для которого указан путь параметром *Path*.

append(F) – открытие файла для добавления в конец его информации.

rename(F, newname) – переименование внешнего файла, с которым связана файловая переменная, на имя, заданное параметром *newname* типа string.

rmdir(Path) – удаление пустого каталога, для которого указан путь параметром *Path* типа string.

Seek (F,Num) – устанавливает указатель открытого файла на компоненту файла с номером *Num* типа Longint. *F* – любая файловая переменная, кроме типа Text.

Функции

eof(F): Boolean – функция определения конца файла. Принимает значение True, если указатель находится за последней компонентой файла. В противном случае принимает значение False.

filesize(F): Longint – возвращает текущий размер файла в компонентах. *F* – любая файловая переменная, кроме типа Text.

Filepos(F): Longint – возвращает номер текущей компоненты файла. *F* – любая файловая переменная, кроме типа Text.

ioresult – результат последней операции ввода-вывода. Принимает значение 0, если ввод-вывод проведен успешно, другое число – в противном случае.

eoln(F) – поиск конца строки файла.

seekeof(F) – поиск конца файла.

При работе со строковыми данными необходимо указывать длину переменной типа *String* в операторе описания типов переменных:

```
var      s: array[1..10] of string[12];
```

Пример. Прочитать из файла A.dat значения элементов матрицы A_{ij} (3×3) и записать ее диагональные элементы в файл B.dat в столбец.

```
Program Matr;
var a:array[1..3,1..3] of real;   {массив (3×3) с переменными
                                   вещественного типа}
    i,j:integer;                  {переменные целого типа}
    f1,f2:text;                   {файловые переменные}
Begin
Assign(f1,'a.dat');               {связь переменной f1
                                   с файлом a.dat}
Reset(f1);                        {открытие файла для чтения}
for i:=1 to 3 do begin
  for j:=1 to 3 do
    read(f1,a[i,j]);              {считывание из файла f1 элементов
                                   строки массива a}
  readln(f1);                     {сброс курсора на следующую
                                   строку в файле f1}
    end;
Close(f1);                        {закрытие файла f1}
Assign(f2,'b.dat');               {связь переменной f2
                                   с файлом b.dat}
Rewrite(f2);                      {открытие нового файла f2}
for i:=1 to 3 do
  for j:=1 to 3 do
if i=j then writeln(f2,a[i,j]);   {запись диагональных
                                   элеметов массива a[i,j]
                                   в файл b.dat}
```

```
Close(f2);                    {закрытие файла f2}
End.
```

Задания для самостоятельной работы

1. Составить программу формирования файла типа txt, состоящего из целых чисел.

2. Составить программу, в которой осуществляется ввод строки информации с клавиатуры и сохранение ее в текстовый файл с расширением .txt.

3. Составить программу, которая формирует файл, состоящий из 5 значений типа integer. Прочитайте файл и вычислите сумму его элементов.

4. Составить программу, которая формирует файл, состоящий из неопределенного количества значений типа integer. Для ввода используйте цикл, выход из цикла – значение 999. После записи выведите файл на экран.

5. Составить программу, которая создает файл, состоящий из 10 значений типа real. Вывести содержимое файла на экран.

6. Составить программу, которая создает файл, состоящий из N значений типа integer. Прочитать информацию из файла и вывести на экран только четные элементы.

7. Составить программу, реализующую следующую последовательность действий с файлами текстового формата a1.txt и a2.txt:

– чтение информации из файла a1.txt и запись в файл a2.txt (построчно);

– уничтожение файла a1.txt.

8. Составить программу, реализующую следующую последовательность действий с файлами b1.txt и b2.txt:

– чтение информации из файла b1.txt и запись в файл b3.rtf (построчно);

– чтение информации из файла b2.txt и запись в файл b3.rtf (построчно);

– переименование файла b1.txt в b1.rtf и файла b2.txt в b2.rtf.

Файлы b1.txt, b2.txt, b1.rtf, b2.rtf, b3.rtf – текстового формата.

9. Рассчитать значения функции $Y = \sin^2(x)$ при изменении аргумента x с шагом 0,05 в диапазоне от 0 до 3. Записать в файл F1.txt

значения *x* и *y*. Затем считать из файла F1.txt значения *x* и *y* и вывести их на экран.

10. Составить программу сортировки по возрастанию и убыванию одномерного массива данных, состоящего из 50 чисел, выбранных произвольно. Организовать вывод результатов на экран и записать полученные массивы в два текстовых файла.

Лабораторная работа № 4

РАБОТА С ЗАПИСЯМИ В ПАСКАЛЕ

Цель работы: освоение правил работы с записями в Паскале.

Запись – это структурированный тип данных, состоящий из фиксированного числа логически связанных компонентов одного или нескольких типов.

Описание записи в Паскале осуществляется с помощью служебного слова *Record*, вслед за которым описываются компоненты записи. Завершается описание служебным словом *end*. Компоненты записи (поля записи) определяются именем, вслед за которым через двоеточие указывается тип этого поля. В отличие от массива, компоненты записи могут быть различного типа.

```
type Car = record
      Nomer:integer;          {номер автомобиля }
      Marka:string[20];       {марка автомобиля}
      fio:string[40];         {фамилия, инициалы владельца}
      Adres:string[60]        {адрес владельца}
   end;
```

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

Описание записей возможно и без использования имени типа, например:

```
var Car = record
      Nomer:integer;
      Marka:string[20];
      fio:string[40];
      Adres:string[60]
   end;
```

Обращение к записи в целом допускается только в операторах присваивания, где слева и справа от знака присваивания используются имена записей одинакового типа. Во всех остальных случаях

оперируют отдельными полями записей. Чтобы обратиться к отдельной компоненте записи, необходимо указать имя записи и через точку указать имя нужного поля, например *car.fio*. Компонентой записи может быть запись (подзапись), в таком случае составное имя будет содержать не два, а большее количество имен.

Пример. Составить программу записи в файл списка студентов группы с указанием их фамилии и года рождения, вывода содержимого файла на экран. Исходные данные:

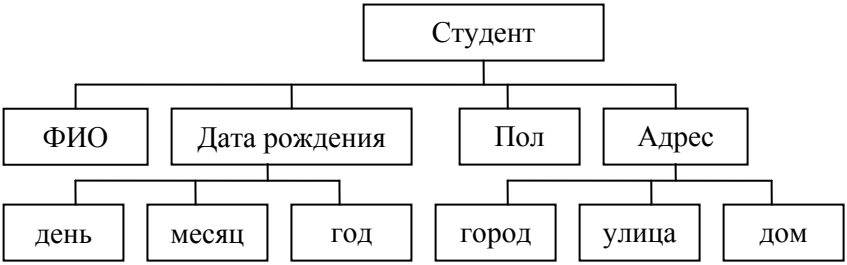
№	ФИО студента	Год рождения
1	Иванов А.С.	1989
2	Петров И.В.	1990

```
Program Spisok;                 {заголовок программы}
   Type
     spis=record                {описание структуры записи}
       fio:String [10];         {поле фамилии на 10 символов}
       gr:integer;              {поле года рождения}
     end;
  Var
    f:file of spis;             {файловая переменная}
    b:spis;                     {переменная типа записи}
    i,n:integer;                {переменные для циклов}

Begin                           {начало операторной части программы}
    Assign(f,'sp.doc');         {привязка файловой переменной
к файлу 'sp.doc'}
    Rewrite(f);                 {открытие файла с именем 'sp.doc'
                                для записи}
    Write('Введите число студентов в группе);
    Readln(n);                  {ввод количества строк}
    Write('Введите ФИО и год рождения  студентов');
     For i:=1 to n do           {цикл для ввода данных}
       begin
         read(b.fio);           {ввод фамилии}
         readln(b.gr);          {ввод года рождения}
         write(f,b);            {сохранение записи в файл}
       end;                     {конец цикла}
```

```
    Close(f);                    {закрытие файла}

    i:=1;                        {установка значения счетчика цикла
                                 на начало}
    writeln;                     {указатель на начало новой строки}
    writeln ('-------------------------------');
    writeln (' № |   ФИО   | ГОД РОЖДЕНИЯ');
    writeln ('-------------------------------');
    Reset(f);                    {открытие файла для чтения}
      While not eof (f) do       {цикл с предусловием определения
                                 конца файла}
        begin
          read(f,b);             {чтение записи из файла}
         writeln(i:2,' |', b.fio:10,'  |    ',b.gr);
                                 {вывод полей записи на экран}
          writeln ('-------------------------------');
i:=i+1;                          {увеличение значения счетчика
                                 для вывода номера строки}
        end;                     {конец цикла}
    writeln ('-------------------------------');
    Writeln('конец ');           {вывод сообщения}
End.                             {конец программы}
```

Задания для самостоятельной работы

1. Составить программу сохранения в файл информации о студентах группы, используя запись со следующей структурой:



Произвести чтение информации из файла и вывести ее на экран.

2. Составить программу, реализующую следующую последовательность действий с файлом Com.zap:

– ввод с клавиатуры двух вещественных чисел и двух значений переменных строкового типа и запись этой информации в файл Com.zap;

– чтение этой информации из файла Com.zap и вывод ее на экран.

Компонентами файла являются записи, состоящие из пары вещественных чисел и пары значений переменных строкового типа:

```
Type ComZap=record
r1, r2:real;
s1, s2:string;
end;
```

3. Описать запись с именем типа Kart, содержащую следующие поля:

– номер измерения (тип integer);

– значение (тип real).

Переменная, определяющая запись, называется Z.

4. Описать запись с именем texn, содержащую информацию о хранящемся на складе техники:

– код техники (тип integer);

– наименование техники (тип string);

– цену (тип real).

Переменная, определяющая запись, имеет название Tovar.

5. Описать запись с именем типа Sportsmeni, содержащую информацию о лучших спортивных достижениях студентов:

– название вида (тип string);

– фамилия рекордсмена (тип string);

– дата установления рекорда (запись Dat, состоящая из полей God, Mes, Den);

– сообщение о результате (real).

Лабораторная работа № 5

ПОДПРОГРАММЫ: ПРОЦЕДУРЫ И ФУНКЦИИ В ПАСКАЛЕ

Цель работы: освоение правил работы с подпрограммами в Паскале.

При создании программ часто требуется выполнять одну и ту же последовательность действий на различных этапах обработки информации. В этих задачах в различных частях встречаются фрагменты, одинаковые по выполняемым действиями, различающихся только в значениях исходных данных. Повторяющаяся группа операторов оформляется в виде самостоятельной программной структуры – подпрограммы. Таким образом, подпрограмма – это самостоятельная часть программы, реализующая определенный алгоритм и допускающая обращение к ней из различных частей основной программы или других подпрограмм. В языке Pascal подпрограммы реализуются в виде процедур и функций.

Все параметры, которые использует подпрограмма, можно разбить на две категории: локальные параметры, используемые только подпрограммой, и глобальные параметры, которые объявляются в основной программе и являются доступными как самой программе, так и всем ее подпрограммам. Также следует различать формальные и фактические параметры. *Формальные* параметры определяются в заголовке подпрограммы в виде списка. При обращении к подпрограмме формальные параметры заменяются на соответствующие *фактические* параметры.

5.1. Процедуры

Процедуры объявляются в описательной части программы. Любая процедура состоит, аналогично программе, из заголовка процедуры и тела процедуры. Заголовок процедуры состоит из служебного слова Procedure, имени процедуры и списка параметров.

Формат описания процедуры выглядит следующим образом:

procedure <имя процедуры>*(*<список формальных параметров>*);*
{раздел описания локальных параметров процедуры, например, констант, типов, меток, переменных}
Begin *{тело процекдуры}*

<оператор 1>;
<оператор 2>;
……

End;

Пример. Процедура поиска суммы чисел:

```
procedure  Sum ( a,b : Real; Var s : Real); {имя процедуры и
список формальных параметров}
        Begin                              {начало процедуры}
            s := a + b ;                   {суммирование чисел}
        End;                               {конец процедуры}
```

В данном примере переменные *a* и *b* являются исходными данными для подпрограммы, *s* – результатом. Поэтому перед *s* в описании параметров подпрограммы стоит слово *Var*.

Вызов такой процедуры производится по ее имени с употреблением фактических параметров:

```
Program Summa;
Var  x, y, z, m, s1, s2 : Real;

procedure  Sum (a,b:Real; var s: Real);
        Begin
            s:=a+b;
        End;

Begin
    Write (' Ввести исходные значения x, y, z ');
    Read (x, y, z, m);
    Sum (x, y, s1);
    Sum (z, m, s2);
    Writeln (' Первая сумма = ', s1);
    Writeln (' Вторая сумма = ', s2);
end.
```

На рисунке 6 представлен алгоритм описанной задачи.

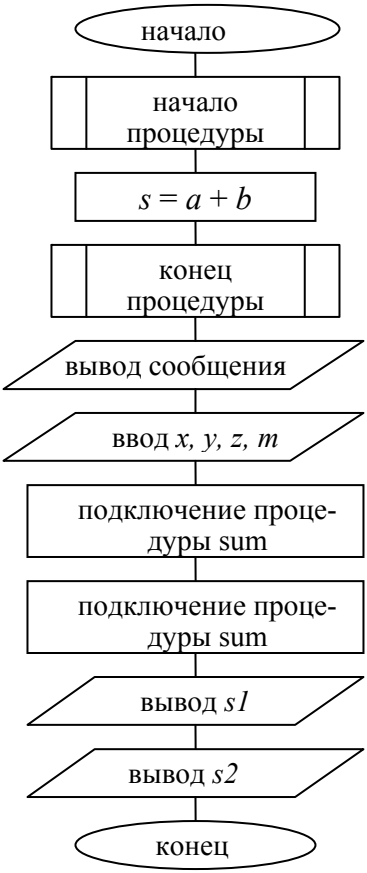


Рисунок 6 – Алгоритм программы

5.2. Функции

Функция – это подпрограмма, результат выполнения которой присваивается ее имени.

Функция является частным случаем подпрограмм. Основное отличие процедуры от функции состоит в том, что результат выполнения функции передается в основную программу как значение имени этой функции, а результаты выполнения процедуры как значения ее параметров.

Функция вызывается в основную программу как встроенная программная стуктура – в выражении записывается ее имя и фактические аргументы, охваченные скобками. Любая функция состоит, аналогично программе, из заголовка функции, раздела объявления параметров (локальных в данном случае) и ее тела. Заголовок функции состоит из служебного слова Function, имени функции и списка параметров, кроме того, указывается тип возвращаемого функцией значения.

Формат описания функции выглядит следующим образом:

function <имя функции>*(*<список формальных параметров>*):*<тип результата>*;*

{раздел описания локальных параметров функции, например, констант, типов, меток, переменных}

Begin

{тело процекдуры}

<оператор 1>;

<оператор 2>;

……

End;

Функция поиска суммы чисел будет выглядеть следующим образом:

```
function  Sum( a,b : Real): Real;   {имя функции со списком
                                     формальных параметров}
          Begin                     {начало функции }
          s := a + b ;              {операция суммирования
                                     двух чисел}
          sum:=s;                   {операция присвоения
                                     результата имени функции}
          End;                      {конец функции }
```

В данном примере переменные *a* и *b* являются исходными (входными) данными для подпрограммы. Перед завершением описания функции необходимо имени функции присвоить результат преобразования. Вызов такой функции производится записью имени функции с указанием фактических параметров.

Пример. Программа вычисления факториалов Fn = n! и Fm = m! с использованием подпрограммы-функции. *{Факториал n! представляет собой произведение n чисел натурального ряда: 1*2*3*...*n.}*

```
Program Factorial;
Var  Fn, Fm, m, n:integer;
      function  Fact (a: integer): longint;   {начало описания функции}
    Var
  p, i:integer;                               {раздел описания
локальных переменных}
 Begin                                        {начало функции}
 p:=1;
 for i:=1 to a do p:=p*i;
fact:=p;
end;                                         {конец функции}
Begin                                        {начало основной программы}
    Write (' Введите значения m, n ');
      Readln (m, n);                         {ввод m, n с клавиатуры}
      Fn:= Fact (n);                         {обращение к функции}
      Fm:= Fact (m);                         {обращение к функции}
      Writeln ('Fn=', Fn:7);                 {вывод результата}
      Writeln ('Fm=', Fm:7);                 {вывод результата}
End.                                         {конец программы}
```

Задания для самостоятельной работы

1. Написать подпрограмму-процедуру для расчета массива из N значений функции $Y(x)$ при изменении аргумента с постоянным шагом h в диапазоне $x1 \ldots x2$, и записи массива в файл. Значения N, $x1$, $x2$ и имя файла задаются с клавиатуры. Организовать вывод массива значений на экран и в текстовый файл. Исходные данные функции $Y(x)$ и параметра h принимаются в соответствии с заданным вариантом (таблица 2).

2. Написать подпрограмму-процедуру для расчета массива значений функции $Y(x)$ для всех значений x из интервала $(a; b)$ с шагом изменения значения аргумента h. Переменные Y_{min} , Y_{max} – минимальное и максимальное значения функции $F(x)$, $\overline{F}$ – среднее значение функции $F(x)$. Исходные данные представлены в таблице 2. Организовать вывод массива значений на экран и в текстовый файл.

3. Напишите подпрограмму-процедуру расчета гипотенузы и площади треугольника по значениям двух катетов, которые вводятся с клавиатуры.

Таблица 2

№ варианта	*Y(x)*	*F(x)*	*h*	*a*	*b*
1	$\frac{\sqrt[3]{\lvert x-1\rvert^5}}{\operatorname{tg}(\lg\lvert x\rvert)}$	–	0,2	–5	–1
2	$e^{x^{(2+x)}}$	–	0,05	5	8
3	$\left\lvert\frac{\sqrt{x-1}+(x-1)^2}{\lvert x-1\rvert}\right\rvert^{2/3}$	–	0,25	4	10
4	$\lvert \sin^3(3x)-2\sin^2(x)-\sin(3x)\rvert$	–	0,01	–2	3
5	$\cos(\sqrt[7]{x})+\sin(\sqrt{\lvert x-e^{x/2}\rvert}$	–	0,4	1	12
6	$\lvert \cos^2(e^x)-2\sin(x)\rvert$	–	0,05	–1	2
7	$\sqrt[5]{\lvert x^3\rvert}+x^2$	–	2	50	100
8	$\ln(\lvert x+2\rvert^{2/3})$	–	0,25	–1	1
9	$\frac{\sqrt[3]{\lvert x-\overline{F}\rvert}}{\sqrt{x}}$	$\cos(x)+2\sin^2(x)$	0,5	2	23
10	$\frac{Y_{\max}}{\sqrt{x^7}}$	$\sin(x^2+4)$	0,02	5	7
11	$\frac{Y_{\min}+\sqrt{\operatorname{tg}(x)}}{24}$	$\sqrt[5]{\operatorname{ctg}(x)}$	0,2	22	26
12	$e^{5x}/(1+\overline{F}^2)$	$\cos^2(x)$	0,25	5	10
13	$\frac{Y_{\max}}{Y_{\min}}x$	$\cos(x)-\sin^2(x)$	0,05	8	12
14	$\frac{Y_{\max}}{\overline{F}}x$	x^3	1	2	20
15	$\frac{\sqrt[3]{\lvert\operatorname{tg}^2(x+2{,}3)\rvert}}{\overline{F}}$	$\sqrt[5]{\ln\lvert x\rvert}$	0,5	–7	1
16	$\overline{F}\cdot x+Y_{\min}$	$\ln(\lvert x-2\rvert)$	0,5	10	15

4. Написать программу с использованием процедуры вывода на экран визитной карточки студента с указанием текущей даты.

5. Напишите программу, состоящую из трех процедур и основной программы. Первая процедура организует ввод двух целых чисел X и Y, вторая вычисляет их сумму, третья выводит результат. Используйте эти процедуры в основной программе.

6. Написать подпрограмму-функцию для расчета суммы значений функции $Y(x)$ для всех значений x из интервала $(a; b)$ с шагом изменения значения аргумента h. Переменные $Y_{\min}, Y_{\max}$ – минимальное и максимальное значения функции $F(x)$. $\overline{F}$ – среднее значение функции $F(x)$ на интервале $(a;b)$. Исходные данные представлены в таблице 2. Организовать вывод суммы на экран и в текстовый файл.

7. Составить программу вычисления функции $Z = a^m$, где m – любое целое (положительное или отрицательное) число, переменная a не равно нулю.

8. Напишите функцию возведения в степень по формуле: $Y=x^n$. Значения переменных x и n вводятся в основной программе с клавиатуры.

9. Вычислить сумму: 1! + 2! + 3! +...+ n!, используя функцию. Значение n вводится с клавиатуры. В основной программе организуйте вывод результата на экран и в файл.

Лабораторная работа № 6

МОДУЛИ В ПАСКАЛЕ

Цель работы: получение навыков работы с внешними программными модулями.

6.1. Создание внешнего модуля

Наличие модулей в Паскале позволяет программировать и проводить отладку программы по частям, создавать библиотеки подпрограмм и данных, а также использовать возможности стандартных модулей. Модуль определяется как программная конструкция (но не программа в смысле возможности непосредственного исполнения), включающая в себя заголовок, интерфейс, исполнительную и инициирующую части.

Заголовок модуля состоит из зарезервированного слова UNIT и идентификатора:

Unit MyModul;

Модуль должен быть помещен в файл, имя которого совпадает с именем модуля, а его расширение должно быть PAS.

Интерфейс модуля предназначен для взаимодействия основной программы с модулем и начинается со служебного слова *Interface.* Интерфейс состоит из раздела описания глобальных имен типов, меток, констант, переменных, а также заголовков процедур, которые могут быть использованы основной программой при вызове этого модуля. В разделе объявления процедур и функций указываются лишь заголовки подпрограмм. Описание подпрограмм приводится в исполнительной части.

Исполнительная часть модуля начинается со служебного слова *Implementation* и содержит полное описание процедур (заголовок, разделы описания и выполнения), а также локальных имен типов, меток, констант и переменных, используемых в инициирующей части.

Инициирующая часть модуля начинается со служебного слова *Begin* и содержит блок операторов, выполняемых при подключении модуля к основной программе. Инициирующая часть может отсут-

ствовать (быть пустой). Заканчивается модуль служебным словом *End* с точкой.

Подключение модулей осуществляется в начале основной программы с помощью служебного слова *Uses* с указанием имен подключаемых модулей, например:

```
Program Prog2;
        Uses MyModul;
```

Приведем пример внешнго модуля и основной программы, в которой он используется:

Внешний модуль: {Файл Unit1.Pas}

```
Unit name;                              {name - имя модуля}
Interface                               {интерфейс модуля}
Var Max, Min real;                      {описание глобальных переменных}
Procedure Sum(a,b:real; var s:real);    {описание заголовка процедуры}
Implemantation                          {исполнительная
                                        часть модуля}
Procedure Sum(a, b:real; var s:real);   {полное описание процедуры}
  begin
s:=a+b;
  end;
Begin                                   {инициирующая часть модуля}
End.
```

Основная программа:

```
Program A;
Uses Unit1;                             {подключение модуля}
Var x,y,z:real;
Begin
Write('Введите два числа через пробел');
Readln(x,y);
Sum(x,y,z);                             {подключение про цедиры,
описанной в модуле}
Writeln('Сумма двух чисел…',z);
End.
```

6.2. Стандартный модуль CRT

Стандартные модули предназначены для расширения возможностей программирования, в том числе не предусмотренных стандартом языка Паскаль. К ним относится, например, модуль CRT, который служит для управления экраном в текстовом режиме, а также для управления клавиатурой и звуковыми сигналами. Модуль содержит библиотеку процедур (подпрограмм) и функций, которые выполняются при их вызове.

При работе с экраном через модуль CRT весь экран разбивается на отдельные строки, а каждая строка – на отдельные позиции, в каждую из которых можно поместить один символ. Таким образом, весь экран разбивается на отдельные неделимые прямоугольные элементы. Для каждого элемента можно задать цвет фона (задний план) и цвет символа (передний план). Кроме того, в случае необходимости символ можно сделать мерцающим.

Модуль CRT позволяет работать не только со всем экраном, но и выделять в нем прямоугольные окна. Любое окно задается своим левым верхним углом и правым нижним углом. Координаты углов задаются двумя координатами: *X* и *Y*. В качестве координаты *X* выступает номер позиции в строке (нумерация начинается с 1 и идет слева направо), а в качестве координаты *Y* – номер строки (нумерация начинается с 1 и идет сверху вниз). При работе в окне координаты отсчитываются от левого верхнего угла окна.

Константы модуля CRT: коды цветов

№	Цвета символов и экрана	№	Цвета символов и экрана
0	Black – черный	8	DarkGray – темно-серый
1	Blue – синий	9	LightBlue – светло-синий
2	Green – зеленый	10	LightGreen – светло-зеленый
3	Cyan – голубой	11	LightCyan – светло-голубой
4	Red – красный	12	LightRed –розовый
5	Magenta – фиолетовый	13	LightMagenta – светло-фиолетовый
6	Brown – коричневый	14	Yellow – желтый
7	LighttGray – светло-серый	15	White – белый
128	Blink– мерцание символа		

Основные процедуры и функции модуля CRT представлены ниже.

1.1. Процедуры задания режимов работы:

TextMode(Mode:Word); – устанавливает текстовый режим, увеличивает текущее окно до целого экрана, устанавливает переменным DirectVideo и CheckSnow значение True. Mode – требуемый режим.

1.2. Процедуры управления цветом:

HighVideo; – устанавливает высокую яркость символов (заменяет цвета 0–7 на цвета 8–15), выводимых далее на экран.

LowVideo; – устанавливает малую яркость символов (заменяет цвета 8–15 на цвета 0–7), выводимых далее на экран.

NormVideo; – устанавливает первоначальную яркость символов, выводимых далее на экран

TextBackground(Color: Byte); – задает цвет фона. Color – задаваемый цвет фона.

TextColor(Color: Byte); – задает цвет символов. Color – задаваемый цвет символов.

1.3. Процедуры работы с экраном:

ClrEol; – удаляет все символы от курсора (включительно) до конца строки, заполняя этот участок строки цветом фона. Цвет фона задается процедурой TextBackground.

ClrScr; – очищает текущее окно, заполняя его цветом фона, и помещает курсор в его верхний левый угол с координатами (1,1). Цвет фона задается процедурой TextBackground.

DelLine; – удаляет строку, в которой находится курсор.

GotoXY(X,Y: Byte); – перемещает курсор к элементу экрана с заданными координатами. *X*, *Y* – координаты элемента экрана (координаты отсчитываются от левого верхнего угла текущего окна).

InsLIne; – выставляет пустую строку на экране в месте расположения курсора и заполняет ее цветом фона. Цвет фона задается процедурой TextBackground.

Window(X1, Y1, X2, Y2 : Byte); – задает размеры окна на экране и помещает курсор в левый верхний угол окна с координатами (1,1). *X1*, *Y1* – координаты левого верхнего угла окна; *X2*, *Y2* – координаты правого нижнего угла окна.

1.4. Функции работы с экраном:

WhereX: Byte; – возвращает текущую координату *X* курсора.

WhereY: Byte; – возвращает текущую координату *Y* курсора.

1.5. Функции работы с клавиатурой:

KeyPressed: Boolean; – анализирует нажатие клавиши клавиатуры (за исключением вспомогательных клавиш - Shift, Alt, NumLock и т.п.). Результат – True, если клавиша нажата, и False – в противном случае.

ReadKey: Char; – считывает символ с клавиатуры и освобождает буфер клавиатуры от считанного символа.

1.6. Процедуры управления звуком:

Sound; – запускает источник звука с частотой Hz герц.

NoSound; – выключает источник звука.

1.7. Процедуры разнообразного назначения:

Delay(Ms: Word); – задает задержку выполнения программы в Ms миллисекунд. Ms – выражение, определяющее величину задержки в миллисекундах.

AssignCrt(var F: Text); – связывает текстовый файл с устройством CRT. *F* – файловая переменная типа Text, связывающаяся с устройством CRT.

Пример процедуры установки цвета текста и фона:

```
procedure My_Color(txt,fon:byte);
begin
 TextColor(txt);
 TextBackGround(fon);
end;
```

6.3. Стандартный модуль GRAPH

Для формирования графических изображений могут использоваться процедуры и функции библиотечного модуля *Graph*. Модуль *Graph* содержит типы, константы, переменные и подпрограммы, позволяющие пользователю создавать изображения с использованием набора различных технических средств для работы с графической информацией. При работе с этими устройствами (адаптерами) весь экран разбивается на отдельные точки – пиксели, которые могут иметь тот или иной цвет. Каждый пиксель имеет две координаты *X* и *Y*. Координата *X* увеличивается по горизонтали слева направо, начиная от нуля, координата *Y* увеличивается по вертикали сверху вниз, начиная от нуля. Количество пикселей зависит от типа адаптера и режима его работы.

Ниже представлены некоторые стандартные процедуры и функции модуля GRAPH:

InitGraph(GraphDriver, GraphMode :Integer; PathToDriver :String); – инициализирует графическую систему, устанавливает графический режим, исходные значения текущего указателя, палитры цвета и т.д.

CloseGraph; – завершает работу в графическом режиме и осуществляет переход в текстовый режим.

MoveTo(X, Y: Integer); – перемещает текущий указатель (курсор) в точку окна с координатами *X*, *Y*. Точка на экране не высвечивается.

LineTo(X, Y: Integer); – проводит линию текущего цвета из текущей позиции в точку с заданными координатами *X*, *Y*. Текущий цвет задается процедурой SetColor, текущие параметры линии – процедурой SetLineStyle.

Line(X1, Y1, X2, Y2: Integer); – проводит линию текущего цвета между точками с координатами *X1*, *Y1* и *X2*, *Y2* без изменения значения указателя координат. Текущий цвет задается процедурой SetColor, текущие параметры линии – процедурой SetLineStyle.

FloodFill(X, Y: Integer; Border: Word); – закрашивает область, ограниченную непрерывной линией, текущим орнаментом и цветом заполнения. Орнамент и цвет заполнения задаются процедурами SetFillStyle или SetFillPattern. *X*, *Y* – координаты любой точки внутри закрашиваемого контура. Border – цвет линии, до которой производится закрашивание.

SetFillStyle(Pattern: word; Color : Word); – задает орнамент и цвет заполнения фигур.

OutTextXY(X, Y,: Integer, TextSt : String); – выводит на экран последовательность символов, начиная с заданных координат *X*, *Y*.

Str(C: Real(Integer), S: String); – преобразует число в последовательность символов.

SetLineStyle(LineStyle: Word; Pattern : Word; Thickness: Word); – задает параметры линии: стиль, шаблон и толщину.

SetTextStyle(Font: Word; Direction: Word; CharSize: Word); – устанавливает тип шрифта, направление текста и размер символов.

В модуле GRAPH используются следующие типы линий для процедуры SetLineStyle:

1. Непрерывная линия – SolidLn (0)
2. Пунктирная линия – Dotteln (1)

3. Штрихпунктирная – CenterLn (2)
4. Штриховая линия – DashedLn (3).

В модуле GRAPH используются следующие константы орнамента для процедуры SetFillStyle:

5. Заполнение цветом фона – EmptyFill (0)
6. Однородное заполнение – SolidFill (1)
7. Заполнение --- – LineFill (2)
8. Заполнение /// – LtSlashFill (3)
9. Заполнение /// толстыми линиями – SlashFill (4)
10. Заполнение \\\ толстыми линиями – BkSlashFill (5)
11. Заполнение \\\ – LtBkSlashFill (6)
12. Заполнение клеткой – HatchFill (7)
13. Заполнение косой клеткой – XHatchFill (8)
14. Заполнение частой сеткой – InterleaveFill (9)
15. Заполнение редкими точками – WideDotFill (10)
16. Заполнение частыми точками – CloseDotFill (11)

Пример использования процедур модуля GRAPH для создания изображения линии:

```
Uses Graph;
 Var
   grDriver: Integer;
   grMode: Integer;
   ErrCode: Integer;
  PathBGI: string;
  Begin
   grDriver := Detect;
  PathBGI:= 'D:\Programming\Bgi';
   InitGraph(grDriver, grMode,PathBGI);
   ErrCode := GraphResult;
   if ErrCode = grOk then
   begin  { Do graphics }
    Line(0, 0, GetMaxX, GetMaxY);
    Readln;
    CloseGraph;
   end
   else
    Writeln('Graphics error:', GraphErrorMsg(ErrCode));
  end.
```

Программа имитации движения окружности внутри окна:

```
uses crt, graph;
VAR
   grDriver: Integer;
   grMode: Integer;
   ErrCode: Integer;
   XM, YM: INTEGER
   xl,xr: integer; (* левая и правая границы окна *)
   yt, yb: integer; (* верхняя и нижняя границы окна *)
   dx,dy,x,y: integer;
begin
      clrscr;
  writeln(' Введите максимальные значения X,У  для центра окружно-
сти');
   readln(xm,ym);
      writeln( 'Введите координаты X левой и правой границ окна');
      readln(xl,xr);
      if (xl >= xr) or  (xl<0) or (xm > xr) or (xm < xl)    then
begin
                writeln(' Координаты заданы неверно');
                halt;
 end;
                writeln('Введите координаты У верхней и нижней границ
окна');
                 readln(yt,yb);
    if (yt >= yb) or  (yt<0) or (ym > yb) or (ym < yt) then
begin
                writeln(' Координаты заданы неверно');
                halt;
end;
  DX:=1;
  DY:=1;
  x:=xl+ trunc((xr-xl)/6);
  y:=yt +trunc((yb-yt)/6);
  clrscr;

 (*************работа  с  графикой ****************)
  grDriver := Detect;
  InitGraph(grDriver, grMode,' ');
  ErrCode := GraphResult;
```

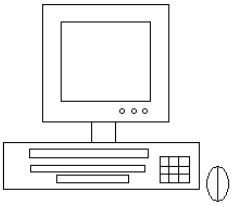
```
if ErrCode = grOk then
begin  { Do graphics }
  LINE(xl,yt,xr,yt);
  LINE(xr,yt,xr,yb);
  LINE(xr,yb,xl,yb);
  LINE(xl,yb,xl,yt);    (* вычертили окно *)
  REPEAT
       setcolor(white);
       CIRCLE(x,y,5);
                (* нарисовали окружность белым цветом *)

 if (x=xl+1) or (x=xr-1) then DX:=-DX;
         if (y=yt+1) or (y=yb-1) then DY:=-DY;
       setcolor(black);
       CIRCLE(x,y,5);

                              (* стерли окружность *)
x:=x+DX;
y:=y+DY;
UNTIL KEYPRESSED;
  Readln;
  CloseGraph;
 end
 else
  Writeln('Graphics error:', GraphErrorMsg(ErrCode));
end.
```

Задания для самостоятельной работы

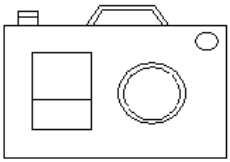
1. Составить программу вывода на экран прямогугольной области, цвет которой изменяется произвольно. Вывод прямоугольника сопровождается звуковым сообщением.

2. Составить программу вывода на экран прямогугольников, цвет и расположение на экране которых изменяется произвольно. Вывод цветных прямоугольных областей сопровождается звуковым сообщением.

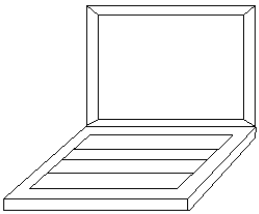
3. С использованием средств стандартного модуля Graph написать программу для вывода на экран рисунка в соответствии с заданным вариантом.



1



2



3

Лабораторная работа № 7

ЧИСЛЕННОЕ ДИФФЕРЕНЦИРОВАНИЕ ФУНКЦИЙ

Цель работы: изучить численные методы дифференцирования функций.

При решении прикладных инженерно-технических задач часто бывает необходимо найти производную определенного порядка от функции $f(x)$, заданной таблично. Возможно, что в силу сложности аналитического выражения функции $f(x)$ непосредственное ее дифференцирование затруднено. В этих случаях обычно используют приближенные численные методы дифференцирования функций.

По определению, первая производная есть предел отношения приращения функции к приращению независимой переменной при стремлении к нулю приращения независимой переменной:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \qquad (1)$$

При вычислении первой производной функции на компьютере бесконечно малое приращение h $(h \to 0)$ заменяют малое, но конечное значение Δx:

$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} + O(\Delta x), \qquad (2)$$

где $O(\Delta x)$ – ошибка вычисления производной, зависящая от Δx.

Приведенная формула называется правой разностной схемой вычисления первой производной. Аналогично может быть записана левая разностная схема:

$$f'(x) = \frac{f(x) - f(x-\Delta x)}{\Delta x} + O(\Delta x). \qquad (3)$$

Для определения ошибки вычисления производной $O(\Delta x)$ функцию $f(x)$ можно разложить в точке $x+\Delta x$ в ряд Тейлора:

$$f(x+\Delta x) = f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2} f''(x) + \frac{\Delta x^3}{6} f'''(x) + \ldots \,. \qquad (4)$$

Отсюда следует, что можно оценить величину ошибки вычисления производной:

$$O(\Delta x) = \frac{\Delta x}{2} f''(x) + \frac{\Delta x^2}{6} f'''(x) + \ldots \,. \qquad (5)$$

Разлагая функцию $f(x)$ в ряд Тейлора в точках $x + \Delta x$ и $x - \Delta x$, т.е. соседних точках от точки x, и вычитая затем один результат от другого, получают центральную разностную схему:

$$f'(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{\Delta x} + O(\Delta x^2), \qquad (6)$$

где $O(\Delta x^2)$ – погрешность вычисления первой производной:

$$O(\Delta x^2) = \frac{\Delta x^2}{6} f'''(x) + \ldots \,. \qquad (7)$$

Таким образом, при численном нахождении производной заменяют отношение бесконечно малых приращений функций и аргумента отношением конечных разностей. Очевидно, что чем меньше будет приращение аргумента, тем точнее численное значение производной. Представленные выше приближенные численные методы называют в программировании двухточечными методами вычисления производных.

Пример. Для функции $y = e^{-x^2}$ необходимо произвести поиск максимального и минимального значений (y_{min}, y_{max}) на заданном интервале $x \in (A;B)$ и соответствующих им значений аргумента (x_{min}, x_{max}); а также максимального и минимального значений первой производной $f'(x)$ функции (y'_{min}, y'_{max}) и соответствующих им значений аргумента (xy'_{min}, xy'_{max}). Шаг изменения аргумента $\Delta x = (B - A) / N$, где $N = 1.0E + 05$. Принять $A = -1$, $B = 5$.

Решение. Таблица соответствия переменных исходным данным задачи представлена в таблице 3.

Таблица 3

Идентификатор переменной	Описание идентификатора
A	начальное значение переменной
B	конечное значение переменной
N	число разбиений расчетного интервала
x	значение переменной
y	значение функции
MAXy	максимальное значение функции
MINy	минимальное значение функции
xMAXy	значение аргумента, соответствующее максимальному значению функции
xMINy	значение аргумента, соответствующее минимальному значению функции
dx	шаг изменения значения аргумента
p	значение производной от заданной функции
MAXp	максимальное значение производной
MINp	минимальное значение производной
xMAXp	значение аргумента, соответствующее максимальному значению производной
xMINp	значение аргумента, соответствующее максимальному значению производной

Текст программы:

```
program differ;
const
A=-1;
B=5;
N=1.0E+05;
var
MAXy, MINy, xMAXy, xMINy : real;
MAXp, MINp, xMAXp, xMINp:  real;
x, y, p, dx: real;
{вычисление значений функции}
function f(x:real):real;
begin
f:=exp(-sqr(x));
```

```
end;
{вычисление производной}
function df(x,dx:real):real;
begin
df:=(f(x+dx)-f(x-dx))/(2*dx);
end;
{начало основной программы}
begin
x:=A;
dx:=(B-A)/N;
MAXy:=f(A);
MINy:=f(A);
while x<B do begin
x:=x+dx;
y:=f(x);
if y>MAXy then begin MAXy:=y; xMAXy:=x end;
if y<MINy then begin MINy:=y; xMINy:=x end;
end;
x:=A+dx;
MAXp:=df(x,dx);
MINp:=df(x,dx);
while x<B do begin
x:=x+dx;
p:=df(x,dx);
if p>MAXp then begin MAXp:=p; xMAXp:=x  end;
if p<MINp then  begin MINp:=p; xMINp:=x end;
end;
writeln(' MAXy = ', MAXy:0:5,' xMAXy = ', xMAXy:0:3);
writeln(' MINy = ', MINy:0:5,' xMINy = ', xMINy:0:3);
writeln(' MAXp = ', MAXp:0:5,' xMAXp = ', xMAXp:0:3);
writeln(' MINp = ', MINp:0:5,' xMINp = ', xMINp:0:3);
readln;
end.
```

Результаты расчета:

MAXy = 1.00000 xMAXy = 0.000
MINy = 0.00000 xMINy = 5.000
MAXp = 0.85776 xMAXh = -0.707
MINp = -0.85776 xMINp = 0.707

Описание программы:

– в разделе const задаем идентификаторы начального (*A*) и конечного (*B*) значения области определения функции и числа *N*;

– в разделе var (переменная) задаем идентификаторы используемых в тексте программы величин и определяем их тип;

– определяем подпрограммы-функции для вычисления значений $y = e^{-x^2}$ и ее производной;

– задаем начальное значение *x*;

– вычисляем шаг (*dx*) изменения значения аргумента (*x*);

– задаем начальные значения MAXy = *f*(*A*) и MINy = *f*(*A*);

– с помощью оператора цикла (while) находим все возможные значения функции, на заданном отрезке значений аргумента, параллельно определяя во вложенном цикле if ее максимальное (MAXy) и минимальное (MINy) значения и соответствующие им значения аргумента (xMAXy; xMINy);

– задаем значение аргумента *x* = *A* + *dx*;

– для отыскания производной (P), ее максимального (MAXP) и минимального (MINp) значений, а также соответствующих значений аргумента (xMAXp; xMINp) используем оператор цикла с предусловием (while) со вложенным условным оператором if;

– выводим результаты расчета на экран.

Задание для самостоятельной работы

Разработать алгоритм и программу, обеспечивающую выполнение:

1) поиска максимального и минимального значения функции *F(x)* на заданном отрезке и соответствующих им значений аргумента;

2) максимального и минимального значений первой производной *F'(x)* заданной функции в интервале $x \in (X_0; X_k)$ и соответствующих им значений аргумента.

Исходные данные представлены в таблице 4, *n* – число разбиений отрезка $[X_0; X_k]$.

Таблица 4

№ вар-та	*f1*	*f2*	*f3*	X_0	X_k	*n*
1	$\ln(\lvert 2x\rvert)$	$(x+2)^{2/3}$	$\lvert 2\cos^3(x) + 2\sin(2x-2)\rvert$	–1	1	100
2	$\lg(\lvert 3x-2\rvert)$	$1-\sin(2x-3)$	$\lvert \sin^3(3x) - 2\sin^2(2x) + \sin(x)\rvert$	–1	2	50
3	$e^{5x} - x^2$	$1-\cos(x^2)$	$\lvert \cos^3(3x) - 2\cos^2(2x) + \cos(x)\rvert$	–2	1	100
4	$(1+\ln\lvert x\rvert) + x^2$	$2 - 1/e^x$	$\lvert \cos^2(e^x) - 2\sin(x)\rvert$	–1	2	50
5	$\ln\lvert x+3\rvert$	$\lvert x\rvert^{3/5}$	$\lvert \sin^2(e^x) - 3\sin(x^2)\rvert$	–1	1	100
6	$\ln(\lvert x+1\rvert)$	$2-\cos(x^2)$	$\lvert \cos^2(e^x) - 2\cos(x^2)\rvert$	–2	2	50
7	$e^x - x^3$	$1-x^2$	$\lvert \cos(2^x) - 5\sin^2(2-x)\rvert$	–1	1	100
8	$\sin(2x)$	$(x-1)^2$	$\lvert e^x - 2\sin^2(x-2)\rvert$	–2	2	50
9	$1-\cos(x^2)$	$1-\sin(x-3x^2)$	$\lvert 2e^x - 5\sin^2(x-1)\rvert$	–2	1	100
10	$\lg(\lvert 2x\rvert)$	$x^2 - e^x$	$\lvert \cos(e^x) - 2\sin^2(x)\rvert$	–1	2	100
11	$x^{2/3}$	$\sin(\lvert 2x-3\rvert)$	$\lvert \sin^2(x) - 3\cos^3(x)\rvert$	–1	1	50
12	$x^{1/5}$	$\cos(\lvert x-2\rvert)$	$\lvert 2\cos(x) - \sin^2(3-x)\rvert$	–1	2	100
13	$3 - x^3$	$2 - e^{2x}$	$\lvert \cos(x) - 2\sin(x^2) + \sin(x)\cos(x^3)\rvert$	–1	1	50
14	$1 - x^{2/3}$	$1 - x^3$	$\lvert \cos^2(x) - 2\sin(2x)\rvert$	–1	2	100
15	$\sin(x)$	$\lg(\lvert x+2\rvert)$	$\lvert \cos^2(x) - \sin^3(3x) - \sin^5(5x)\rvert$	–2	1	50
16	$2 - x^{3/5}$	$\ln(\lvert x+1\rvert)$	$\lvert \sin^2(2x) - 2\sin(3x)\rvert$	–2	2	100
17	$\sin(x^2+1)$	$2x^3$	$\lvert \sin^2(3x) - 2\cos(2x) - \sin(x)\rvert$	–1	1	50
18	$2x^3$	$\sin(2x-1)$	$\lvert \cos^2(3x) + 2\sin(2x-1) - 2\sin(x)\rvert$	–2	2	100
19	$1 - e^{2x}$	$\cos(3x-1)$	$\lvert \sin^3(3x-2) + \sin(2x-1) - \sin(x)\rvert$	–1	1	50
20	$x^{1/4}$	$\ln(\lvert 2x-1\rvert)$	$\lvert \sin^3(3x) - \cos(x) - 2\sin(x)\rvert$	–1	2	100

Лабораторная работа № 8

ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ

Цель работы: изучить численные методы вычисления определенных интегралов.

Под численным интегрированием понимают вычисление значения определенного интеграла с помощью численных (как правило, приближенных) методов. Численное интегрирование применяется в тех случаях, когда подынтегральная функция не задана аналитически (например, в виде таблицы) либо первообразная функции не может быть выражена аналитически, либо вид первообразной настолько сложен, что быстрее вычислить значение интеграла численным методом, чем по формуле Ньютона–Лейбница.

Метод прямоугольников. Пусть на отрезке $[a; b]$ задана непрерывная функция $y = f(x)$ и требуется вычислить определенный интеграл:

$$\int_a^b f(x)dx. \qquad (1)$$

Разделим отрезок $[a; b]$ точками $a = x_0, x_1, x_2, \ldots, x_n = b$ на n равных частей длины Δx, где $\Delta x = (b - a) / n$.

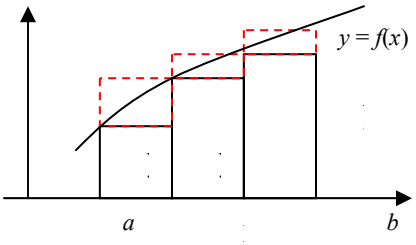


Рисунок 7 – Схема расчета определенного интеграла численными методами (левых и правых прямоугольников)

Обозначим через y_0, y_1, y_2, … ,y_{n-1}, y_n значение функции $f(x)$ в точках x_0, x_1, x_2, … , x_n. Далее подынтегральную функцию заменяем функцией, которая имеет ступенчатый вид и составим суммы: $y_0\Delta x + y_1\Delta x + y_2\Delta x + \ldots + y_{n-1}\Delta x$; $y_1\Delta x + y_2\Delta x + \ldots + y_n\Delta x$.

Каждое слагаемое этих сумм выражает площадь полученных прямоугольников с основанием Δx, которое является шириной прямоугольника, и длиной, выраженной через y_i: $S_{\text{пр}} = y_i\Delta x$. Каждая из этих сумм является интегральной суммой для $f(x)$ на отрезке $[a, b]$, и равна площади ступенчатых фигур и приближенно выражает интеграл. Вынесем $\Delta x = (b - a)/n$ из каждой суммы:

$$\int_a^b f(x)dx \approx \Delta x(y_0 + y_1 + \ldots + y_{n-1}); \tag{2}$$

$$\int_a^b f(x)dx \approx \Delta x(y_1 + y_2 + \ldots + y_n). \tag{3}$$

Выразив x, получим окончательно:

$$\int_a^b f(x)dx \approx ((b-a)/n)(y_0 + y_1 + \ldots + y_{n-1}); \tag{4}$$

$$\int_a^b f(x)dx \approx ((b-a)/n)(y_1 + y_2 + \ldots + y_n). \tag{5}$$

Это и есть формулы прямоугольников. Их две, так как можно использовать два способа замены подынтегральной функции. Если $f(x)$ – положительная и возрастающая функция, то формула (3) выражает S фигуры, расположенной под графиком, составленной из входящих прямоугольников, а формула (5) – площадь ступенчатой фигуры, расположенной под графиком функции, составленной из выходящих треугольников.

Ошибка, совершаемая при вычислении интегралов по формуле прямоугольников, будет тем меньше, чем больше число n (то есть, чем меньше шаг деления). Шаг деления равен:

$$\Delta x = \frac{b-a}{n}. \qquad (6)$$

Результат, полученный по формуле (4), заведомо дает большую площадь прямоугольника, а по формуле (5) дает заведомо меньшую площадь, для получения среднего результата используется формула средних прямоугольников:

$$\int f(x)dx = \frac{b-a}{n}(y_0 + y_1 + ... + y_{n-1}). \qquad (7)$$

Метод трапеций. При вычислении интеграла с помощью формулы трапеций подынтегральная функция $f(x)$ заменяется функцией, график которой представляет собой ломанную линию, звенья которой соединяют концы ординат y_{i-1} и y_i.

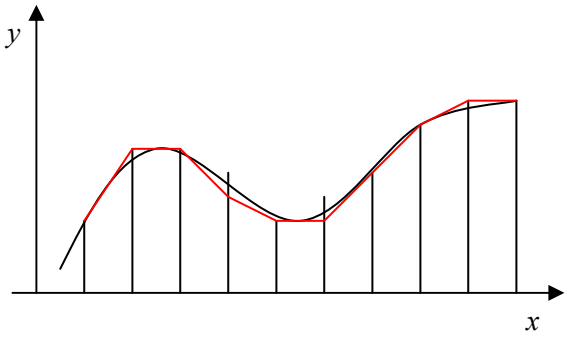


Рисунок 8 – Схема расчета определенного интеграла методом трапеций

Площадь криволинейной трапеции, ограниченной линиями $x = a$, $x = b$, $y = 0$, $y = f(x)$, (следуя из геометрического смысла), приблизительно равна сумме площадей обычных трапеций с основаниями y_{i-1} и y_i и высотой $h = (b - a)/n$, т.е. в математическом виде:

$$\int_a^b f(x)dx \approx \frac{y_0+y_1}{2}h+\frac{y_1+y_2}{2}h+\ldots+\frac{y_{n-1}+y_n}{2}h=h(\frac{y_0+y_1}{2}+\frac{y_1+y_2}{2}+\ldots+\frac{y_{n-1}+y_n}{2})=$$

$$=\frac{b-a}{n}\sum_{i=1}^{n}\frac{y_{i-1}+y_i}{2} \tag{8}$$

$$\int_a^b f(x)dx \approx h(\frac{y_0+y_1+y_1+y_2+\ldots+y_{n-1}+y_n}{2})=$$

$$=h(\frac{y_0+2y_1+2y_2+\ldots+2y_{n-1}+y_n}{2})=h(\frac{y_0+y_n}{2}+\frac{2y_1}{2}+\frac{2y_2}{2}+\ldots+\frac{2y_{n-1}}{2})=$$

$$=h(\frac{y_0+y_n}{2}+y_1+y_2+\ldots+y_{n-1})=\frac{b-a}{n}(\frac{y_0+y_n}{2}+y_1+y_2+\ldots+y_{n-1})$$

Формула (9) и есть формула трапеций.

$$\int_a^b f(x)dx=\frac{b-a}{n}(\frac{y_0+y_n}{2}+y_1+y_2+\ldots+y_{n-1}). \tag{9}$$

Метод Симпсона (формула парабол). Разделим отрезок [a;b] на четное число равных частей $n = 2m$. Площадь криволинейной трапеции, соответствующей первым двум отрезкам [x_0; x_1], [x_1; x_2] и ограниченной заданной кривой $y = f(x)$, заменим площадью криволинейной трапеции, которая ограничена параболой второй степени, проходящей через три точки $M_0[x_0; y_0]$, $M_1[x_1; y_1]$, $M_2[x_2; y_2]$ и имеющей ось, параллельную оси Oy. Такую криволинейную трапецию будем называть параболической трапецией.

Уравнение параболы с осью, параллельной оси Oy, имеет вид:

$$y = Ax^2 + Bx + C. \tag{10}$$

Коэффициенты A, B и C однозначно определяются из условия, что парабола проходит через три заданные точки. Аналогичные параболы строятся и для других пар отрезков. Сумма параболических трапеций и даст приближенное значение интеграла.

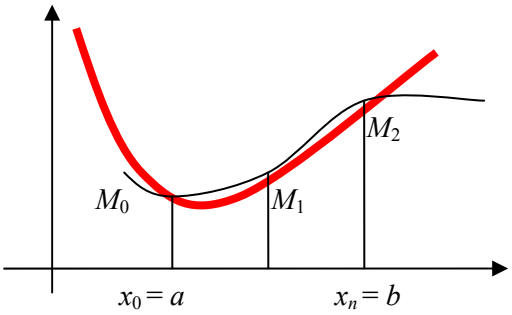


Рисунок 9 – Схема расчета определенного интеграла методом Симпсона

Площадь одной параболической трапеции, ограниченной параболой $y = Ax^2 + Bx + C$, осью Ox и двумя ординатами, расстояние между которыми равно $2h$, равна

$$S = \frac{h}{3}(y_0 + 4y_1 + y_2), \tag{11}$$

где y_0 и y_2 – крайние ординаты;

y_1 – ордината кривой в середине отрезка.

Можно написать приближенные равенства, учитывая, что

$$h = \Delta x = \frac{b-a}{2m}; \tag{12}$$

$$\int_{a=x_0}^{x_2} f(x)dx \approx \frac{\Delta x}{3}(y_0 + 4y_1 + y_2); \tag{13}$$

$$\int_{x_2}^{x_4} f(x)dx \approx \frac{\Delta x}{3}(y_2 + 4y_3 + y_4); \tag{14}$$

$$\int_{2m-2}^{x_{2m}=b} f(x)dx \approx \frac{\Delta x}{3}(y_{2m-2} + 4y_{2m-1} + y_{2m}). \tag{15}$$

Складывая левые и правые части, получим слева искомый интеграл, справа его приближенное значение:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{3}((y_0 + 4y_1 + y_2) + (y_2 + 4y_3 + y_4) + ... + (y_{2m-2} + 4y_{2m-1} + y_{2m})) =$$
$$= \frac{\Delta x}{3}(y_0 + 4y_1 + y_2 + y_2 + 4y_3 + y_4 + ... + y_{2m-2} + y_{2m-1} + y_{2m})) =$$
$$= \frac{\Delta x}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + ... + y_{2m-2} + y_{2m-1} + y_{2m}) \qquad (16)$$

или

$$\int_a^b f(x)dx \approx \frac{\Delta x}{3}(y_0 + y_{2m} + 2(y_2 + y_4 + ... + y_{2m-2}) + 4(y_1 + y_3 + ... + y_{2m-1}))$$
$$\int_a^b f(x)dx = \frac{b-a}{6m}(y_0 + y_{2m} + 2(y_2 + y_4 + ... + y_{2m-2}) + 4(y_1 + y_3 + ... + y_{2m-1})) \qquad (17)$$

Это и есть формула Симпсона.

Пример. Вычислить площадь фигуры, ограниченной сверху кривой $y = e^{-x^2}$, снизу осью абсцисс, слева прямой $x_0 = 0$ и справа прямой $x_k = 2$. Вычисление площади (определенного интеграла) проводить методом трапеций, при числе разбиений отрезка $n = 200$.

Текст программы:

```
Program Integral;
Const
{границы интегрирования}
A=0; B=2;
{число разбиений}
N=200;
{расчет значений функции}
Function F(x:Real):Real;
Begin
F:=exp(-sqr(x));
End;
```

```
{вычисление определенного интеграла}
Function IntTRP(x0,xk:Real; k:Word):Real;
Var
x,dx,Sum:Real;
i:Word;
Begin
x:=x0;
dx:=(xk-x0)/k;
Sum:=(F(x0)+F(xk))/2;
for i:=1 to k-1 do
begin
x:=x+dx;
Sum:=Sum+F(x);
end;
IntTRP:=Sum*dx;
End;
{начало основной программы}
Begin
writeln(' IntTRP(A,B,k)= ',IntTRP(A,B,N):0:5);
readln;
End.
```

Результат:

K = 200 IntTRP(A,B,N) = 0.88208

Задание для самостоятельной работы

Разработать алгоритм и программу, обеспечивающую вычисление площади фигуры, ограниченной сверху кривой $y = f(x)$, снизу осью абсцисс, слева прямой $x = -1$ и справа прямой $x = 2$. Исходные данные представлены в таблице 5 (n – число разбиений отрезка $[X_0; X_k]$).

Таблица 5

№ вар-та	f_1	f_2	f_3	X_0	X_k	n
1	$\ln(\lvert x-2\rvert)$	$(x+2)^{2/3}$	$\lvert 2\cos^3(x)+2\sin(2x-2)\rvert$	–1	1	100
2	$\lg(\lvert x+2\rvert)$	$1-\sin(2x-3)$	$\lvert \sin^3(3x)-2\sin^2(2x)+\sin(x)\rvert$	–1	2	200
3	$e^{2x}-x^2$	$1+\cos(x^2)$	$\lvert \cos^3(3x)-2\cos^2(2x)+\cos(x)\rvert$	–2	1	100
4	$1+\ln(\lvert x\rvert)$	$2+1/e^x$	$\lvert \cos^2(e^x)-2\sin(x)\rvert$	–1	2	200
5	$\ln\lvert x-3\rvert$	$x+\lvert x\rvert^{3/5}$	$\lvert \sin^2(e^x)-3\sin(x^2)\rvert$	–1	1	100

Окончание таблицы 5

№ вар-та	f_1	f_2	f_3	X_0	X_k	n
6	$\exp(\|2x\|)$	$\ln(\|x+2\|^{2/3})$	$\|\ 2\cos^3(x)+2\sin(2x-2)\ \|$	–1	1	100
7	$\exp(\|3x-2\|)$	$\exp(1-\sin(2x-3))$	$\|\ \sin^3(3x)-2\sin^2(2x)++\sin(x)\ \|$	–1	2	200
8	$\sin(e^{5x}-x^2)$	$\exp(1-\cos(x^2))$	$\|\ \cos^3(3x)-2\cos^2(2x)+\cos(x)\ \|$	–2	1	100
9	$\sin(1+\ln\|x\|)$	$\cos(2-1/e^x)$	$\|\ \cos^2(e^x)-2\sin(x)\ \|$	–1	2	200
10	$\exp(\|x+3\|)$	$\exp(x^{3/5}-2x)$	$\|\ \sin^2(e^x)-3\sin(x^2)\ \|$	–1	1	100
11	$5+x^5$	$\ln(\|2x-3\|)$	$\|\ \cos^2(x)-3\sin^3(x)\ \|$	–1	1	200
12	$2+x^{1/3}$	$\lg(\|x-2\|)$	$\|\ 2\sin(x)-\sin^2(3-x)\ \|$	–1	2	100
13	$3-x^3$	$2+e^{2x}$	$\|\ \cos(x)-2\sin(x^2)+\sin(x)\cos(x^3)\|$	–1	1	200
14	$1+x^{2/3}$	$1+x^3$	$\|\ \cos^2(x)-2\sin(2x)\ \|$	–1	2	100
15	$e^{\sin(x)}$	$\lg(\|x+2\|)$	$\|\ \cos^2(x)-\sin^3(3x)-\sin^5(5x)\ \|$	–2	1	200
16	$2+x^{3/5}$	$\ln(\|x+1\|)$	$\|\ \sin^2(2x)-2\sin(3x)\ \|$	–2	2	100
17	$\lg(x^2+1)$	$5x^3$	$\|\ \sin^2(3x)-2\cos(2x)-\sin(x)\ \|$	–1	1	200
18	$2x^3$	$\sin(2x-1)$	$\|\ \cos^2(3x)+2\sin(2x-1)-2\sin(x)\ \|$	–2	2	100
19	$1-e^{2x}$	$\cos(3x-1)$	$\|\ \sin^3(3x-2)+\sin^2(2x-1)-\sin(x)\ \|$	–1	1	200
20	$2+x^{1/4}$	$\lg(\|2x-1\|)$	$\|\ \sin^2(3x)-\cos(x)-2\sin(x)\ \|$	–1	2	100
21	$1-\ln(\|2x\|)$	$2x-(x+2)^{2/3}$	$\|\ \sin^2(3x)-3\cos(x)-\sin(x)\ \|$	–1	1	100
22	$\lg(\|x-2\|)$	$2\sin^2(x)$	$\|\ \cos^3(x)+2\sin(x-1)\ \|$	–1	2	200
23	$e^{5x}/(1+x^2)$	$\cos^2(x)$	$\|\ \sin^3(3x)-2\sin^2(2x)-\sin(3x)\ \|$	–2	1	100
24	$(1-e^x)/(1+x^2)$	$(e^x+2)/(2+x^{1/2})$	$\|\ \cos^3(3x)-\cos^2(2x)+\cos(2x)\ \|$	–1	2	200
25	$\ln(\|x-2\|)$	$\|x+1\|^{3/5}$	$\|\ 2\cos^2(e^x)-\sin(x)+\cos(x)\ \|$	–1	1	100
26	$\ln(\|x+1\|)$	$\cos(x/3)$	$\|\ \sin^2(e^x)-3\sin(x^2)+\sin(x)\|$	–2	2	200
27	x^3-1/x	$x^3+1/(x+2)$	$\|\ \cos^2(e^x)-2\cos(x^2)+\sin(x)\|$	–1	1	100
28	$\sin(2x/3)$	$1/(x+5)$	$\|\ \cos(2^x)-5\sin^2(2-x)+\sin(x)\ \|$	–2	2	200
29	$1-\cos(x^2)$	$2x^3+3x^2$	$\|\ e^x-2\sin^2(x-2)+\cos(x)\ \|$	–1	1	100
30	$\lg(\|2x-5\|)$	x^2-e^x	$\|\ 2\cos^3(x)+2\sin(2x-2)-\cos(x)\ \|$	–1	2	200

ЛИТЕРАТУРА

1. Пискунов, Н. С. Интегральное и дифференциальное исчисления для втузов: учебное пособие для втузов: в 2 т. / Н. С. Пискунов. – М.: Наука. 1985.

2. Школа работы на IBM PC. Ч. 2 [Электронный ресурс] / М. Е. Сидоров, О. В. Трушин. – Режим доступа: http://www.nullsoft.ru/tv/pascal.htm.

3. Рапаков, Г. Г. Turbo Pascal для студентов и школьников / Г. Г. Рапаков, С. Ю. Ржеуцкая. – СПб.: БХВ-Петербург, 2005. – (Основы информатики).

4. Турбо Паскаль 7.0 / под ред. В. Кораблева. – 16-е изд. – СПб. [и др.]: BHV, 2004. – (Самоучитель).

5. Информатика: лабораторный практикум для студентов специальности 1-42 01 01 «Металлургические процессы и материалообработка»: в 2 ч. / сост.: И. В. Рафальский, А. В. Арабей. – Минск: БНТУ, 2009. – Ч. 1.

6. Рафальский, И. В. Учебно-методическое пособие по дисциплине «Информатика» для студентов специальности Т.02.02.00 «Технология, оборудование и автоматизация обработки материалов» / И. В. Рафальский, Н. П. Юркевич, А. В. Мазуренок. – Минск: БГПА, 2001. – 85 с.

СОДЕРЖАНИЕ

Учебное издание

ИНФОРМАТИКА

Лабораторный практикум
для студентов специальности
1-42 01 01 «Металлургическое производство
и материалообработка»

В 2 частях

Часть 2

Составители:
РАФАЛЬСКИЙ Игорь Владимирович
АРАБЕЙ Анастасия Витальевна

Технический редактор *Д. А. Исаев*
Компьютерная верстка *Д. А. Исаева*

Подписано в печать 29.10.2013. Формат 60×84 $^{1}/_{16}$. Бумага офсетная. Ризография.
Усл. печ. л. 4.01. Уч.-изд. л. 3,14. Тираж 100. Заказ 623.
Издатель и полиграфическое исполнение: Белорусский национальный технический университет. ЛИ № 02330/0494349 от 16.03.2009. Пр. Независимости, 65. 220013, г. Минск.