БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

А.Н. Чичко
О.А. Сачек
О.И. Чичко

ИНФОРМАТИКА. ПРАКТИКУМ

Допущено Министерством образования Республики Беларусь в качестве учебного пособия для студентов высших учебных заведений по техническим специальностям

Минск
БНТУ
2011

УДК 002.6(075.8)
ББК 32.818я7
Ч 72

Рецензенты:
кандидат технических наук, доцент, заведующий кафедрой программного обеспечения интеллектуальных и компьютерных систем, Гродненского государственного университета имени Янки Купалы *В.Г. Родченко*;
кандидат физико-математических наук, профессор кафедры прикладной математики и информатики, Белорусского государственного педагогического университета имени Максима Танка *А.И. Павловский*

Чичко, А.Н.

Ч 72 Информатика. Практикум: учебное пособие / А.Н. Чичко, О.А. Сачек, О.И. Чичко. – Минск: БНТУ, 2011. – 399 с.

ISBN 978-985-525-566-7.

Настоящее учебное пособие содержит задачи с решениями, рассматриваемые в рамках программы вузовского курса по информационным технологиям. В нем представлены пошаговые алгоритмы, блок-схемы и программы различных по уровню сложности математических задач. Отличительной чертой издания является наличие подробных решений, а также большое количество оригинальных примеров, позволяющих читателю самостоятельно освоить процесс разработки алгоритма и его реализацию в среде программирования Turbo Pascal. Большое количество предлагаемых вариантов задач делает данное издание эффективным инструментом при организации индивидуальной работы со студентами. Будет полезно студентам очной и заочной форм обучения, желающим быстро научиться разрабатывать алгоритмы и программы, а также слушателям, проходящим курс переподготовки по дисциплинам, связанным с информационными технологиями.

УДК 002.6 (075.8)
ББК 32.818 я7

ISBN 978-985-525-566-7

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современный уровень требований к инженерам технического профиля предполагает хорошее знание ими методов алгоритмизации и решения сложных математических задач. Современный инженер должен уметь перевести техническую задачу в алгоритмическое представление, на основе которого ее можно решить. Составление алгоритма – самый сложный этап творчества, часто вызывающий трудности у студентов, изучающих курс информационных технологий. Поэтому в данном учебном пособии рассмотрена целая группа алгоритмов, начиная от простых и заканчивая сложными, которые связаны с компьютерными методами и технологиями. В качестве инструментария для реализации рассматриваемых алгоритмов использованы MS Excel и Turbo Pascal. Несмотря на то, что по дисциплине «Информационные технологии» имеется большое количество учебной литературы, практикумы с подробными решениями задач по информационным технологиям являются скорее исключением, чем правилом. Опыт работы авторов со студентами показывает необходимость таких учебных пособий, которые позволяют без помощи преподавателя научиться разрабатывать алгоритмы и решать задачи хотя бы на одном языке программирования. Приобретя основные навыки по этим вопросам, обучающийся способен сложнейшие задачи осваивать самостоятельно.

Отличительной чертой пособия является множество упражнений и задач с решениями, что позволяет студентам самостоятельно освоить многие вопросы информационных технологий. В пособии представлено около 50 вариантов различного класса математических задач, что будет полезным для преподавателя, так как позволяет наиболее эффективно организовать индивидуальную работу со студентами. Многие задачи рассмотрены настолько подробно, что становятся полезными для слушателей как дневного, так и заочной форм обучения. Учебное пособие может быть использовано для студентов, занимающихся в учреждениях, использующих дистанционную форму обучения по дисциплинам, связанным с информационными технологиями.

Учебное пособие состоит из пяти основных разделов, каждый из которых несет свою смысловую нагрузку. В первом разделе основной акцент сделан на разработку простейших алгоритмов и про-

грамм, которые являются введением в процесс алгоритмизации. Во втором разделе представлены сложные математические задачи по составлению алгоритмов и программ, которые являются оригинальными. В третьем и четвертом разделах рассмотрены численные методы решения задач на персональном компьютере (ПК) с использованием языка программирования Pascal и возможностей MS Excel, причем подробные решения позволяют неподготовленному читателю освоить основные методы вычислений, используемые в прикладной математике. В пятом разделе представлены индивидуальные задания, которые могут быть использованы преподавателем в курсовом и дипломном проектировании.

В приложениях пособия приведены краткие сведения, необходимые для освоения численных методов, разработки алгоритмов (правила и примеры построения, алгоритм процесса построения алгоритмов с помощью MS Word и MS Visio) и программ (операторы, процедуры и функции языка Pascal, приемы работы в среде Turbo Pascal), дающие читателю математические основы для решения задач курса по информационным технологиям.

1. ЗАДАЧИ ПО РАЗРАБОТКЕ АЛГОРИТМОВ И ПРОГРАММ

1.1. **Линейные вычислительные процессы**

Задача 1.1. Разработать схему алгоритма и программу для вычисления площади треугольника S по заданным с клавиатуры значениям стороны a и высоты h, проведенной к этой стороне. Значение площади вывести на экран.

Решение задачи

Алгоритм решения задачи состоит в вычислении площади треугольника по введенным значениям стороны и высоты. Логика алгоритма сводится к построению алгоритма с линейной структурой, в котором каждая последующая процедура выполняется после предыдущей. Вычисление площади треугольника производится по формуле $S = \frac{a \cdot h}{2}$.

Разработанная схема алгоритма представлена на рис. 1.1. Изображенные на рисунке символы пронумерованы.

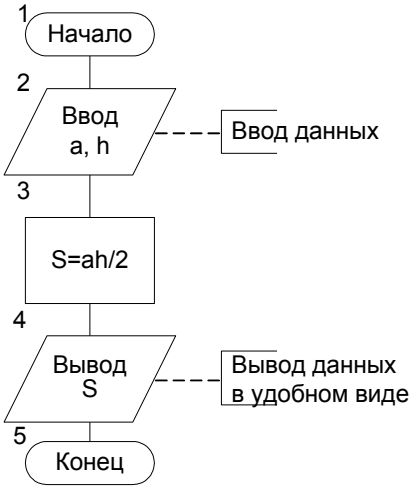


Рис. 1.1

Обозначения схемы алгоритма

S – значение площади треугольника

Программа

```
Program Prog1_1;
Var
   a, h, S : Real;  {описание переменных программы}
Begin               {начало раздела операторов программы}
   Write('Введите значение стороны a ');
   Readln(a);       {чтение значения стороны a}
   Write('Введите значение высоты h ');
   Readln(h);       {чтение значения высоты h}
   S := a*h/2;      {вычисление площади}
   Writeln('Площадь треугольника S = ', S:7:2);{вывод на экран}
End.                {конец программы}
```

Результат работы программы

```
Введите значение стороны a 1.5
Введите значение высоты h 7.2
Площадь треугольника S = 5.40
```

1.2. **Циклические вычислительные процессы**

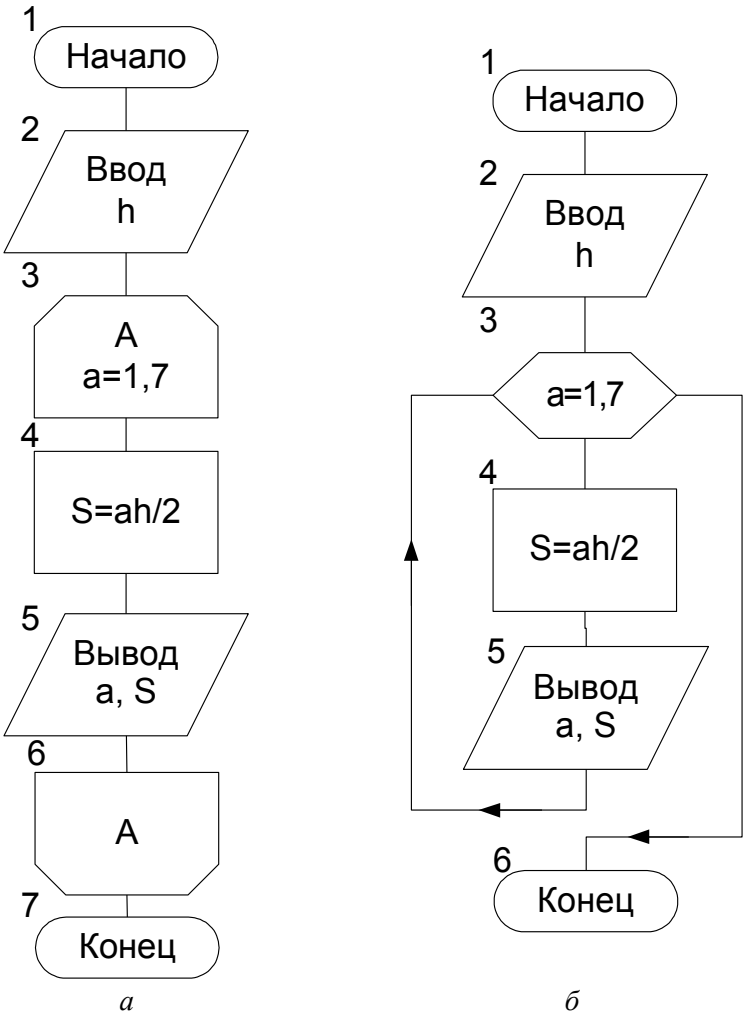
***Задача 1.2*.** Разработать схему алгоритма и программу вычисления и вывода на экран значений площадей треугольников с различными основаниями a и постоянной высотой h, проведенной к основанию. Величина a принимает значения $a = 1; 7$ (1). Значение высоты h вводится с клавиатуры.

Решение задачи

Алгоритм решения задачи состоит в последовательном переборе значений a от 1 до 7 с шагом 1 при вычислении площади по формуле $S = \frac{a \cdot h}{2}$. Алгоритм решения задачи состоит из одного цикла и представлен на рис. 1.2 в двух вариантах:

1) с использованием парного символа «Граница цикла» (рис. 1.2, *а*);

2) с использованием символа «Подготовка» (рис. 1.2, *б*).



а *б*

Рис. 1.2

Обозначения схем алгоритма

S – значение площади треугольника.

Программа

```
Program Prog1_2;
Var
   h, S : Real;
   a : Byte;
Begin                  {начало раздела операторов программы}
   Write('Введите значение высоты h ');
   Readln(h);          {чтение значения высоты h}
   for a:=1 to 7 do    {цикл для перебора значений основания a}
   begin               {начало цикла}
      S := a*h/2;      {вычисление площади S}
      Writeln('При a = ', a, ' S = ', S:7:2);{вывод результатов}
   end;                {конец цикла};
End.                   {конец программы}
```

Результат работы программы

```
Введите значение высоты h 7.2
При a = 1 S =  3.60
При a = 2 S =  7.20
При a = 3 S = 10.80
При a = 4 S = 14.40
При a = 5 S = 18.00
При a = 6 S = 21.60
При a = 7 S = 25.20
```

***Задача** 1.3.* Разработать схему алгоритма и программу вычисления и вывода на экран значений площадей треугольников по изменяющимся значениям сторон и высот. Значение стороны a изменяется по закону $a = 1; 3\ (1)$, высота h проведена к стороне a и изменяется по закону $1; 5\ (0{,}8)$.

Решение задачи

В алгоритме этой задачи необходимо использовать два цикла. Для последовательного перебора значений a используется цикл от 1 до 3 с шагом 1, а для перебора значений h используется цикл от 1 до 5 с шагом 0,8 с предусловием $h \le 5$. На рис. 1.3 представлена схема алгоритма решения задачи с использованием различных символов цикла, а именно: цикл для перебора переменной h организован в виде разветвляющейся структуры, а для перебора переменной a – в

виде циклической структуры с использованием парного символа «Граница цикла».

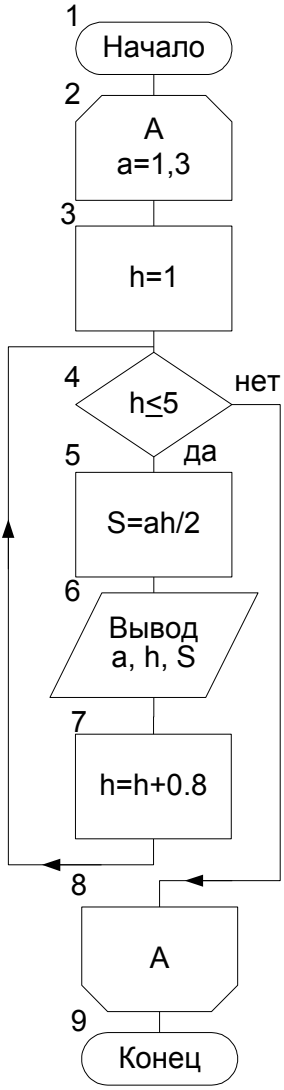


Рис. 1.3

Обозначения схем алгоритма

S – значение площади треугольника.

Программа

```
Program Prog1_3;
Var
   h, S : Real;
   a : Byte;
Begin                {начало раздела операторов программы}
   for a := 1 to 3 do
   Begin                    {начало цикла 1}
      h := 1;               {начальное значение высоты h}
      While h <= 5 do       {выполнять цикл пока h <= 5}
      Begin                 {начало цикла 2}
         S := a*h/2;        {вычисление площади S}
         {вывод на экран}
         Writeln('При a = ', a, ' , h = ', h:7:2, ' S = ', S:7:2);
         h := h + 0.8;      {приращение высоты h}
      end;                  {конец цикла 2}
   end;                     {конец цикла 1}
End.                        {конец программы}
```

Результат работы программы

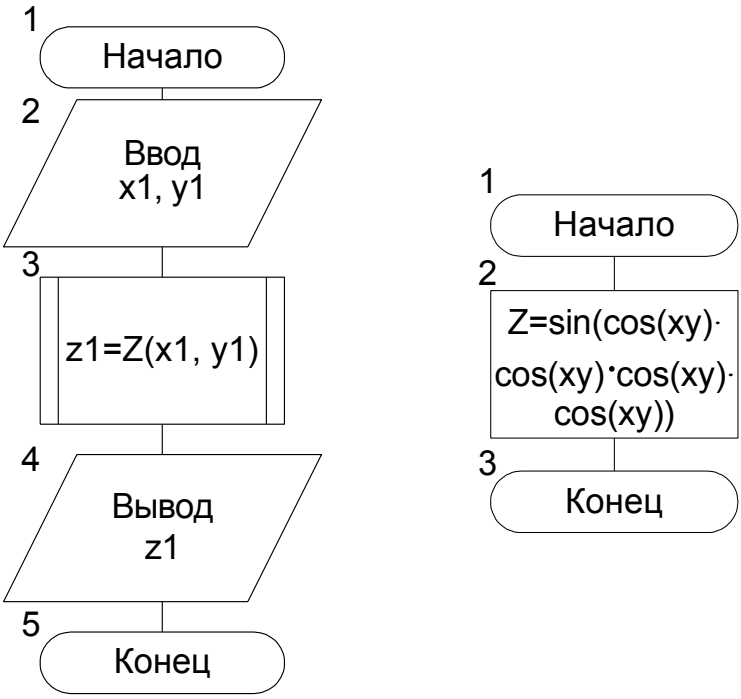
```
При a = 1 h =  1.00 S =  0.50
При a = 1 h =  1.80 S =  0.90
При a = 1 h =  2.60 S =  1.30
При a = 1 h =  3.40 S =  1.70
При a = 1 h =  4.20 S =  2.10
При a = 2 h =  1.00 S =  1.00
При a = 2 h =  1.80 S =  1.80
При a = 2 h =  2.60 S =  2.60
При a = 2 h =  3.40 S =  3.40
При a = 2 h =  4.20 S =  4.20
При a = 3 h =  1.00 S =  1.50
При a = 3 h =  1.80 S =  2.70
При a = 3 h =  2.60 S =  3.90
При a = 3 h =  3.40 S =  5.10
При a = 3 h =  4.20 S =  6.30
```

1.3. **Вычислительные процессы с использованием процедур**

Задача 1.4. Разработать алгоритм и программу для вычисления и вывода на экран значений функции $z = \sin[\cos^4(x \cdot y)]$. Значения переменных *x* и *y* вводятся с клавиатуры. Для решения задачи необходимо использовать подпрограмму (*function*).

Решение задачи

Общий алгоритм решения задачи и схема вычисления функции в подпрограмме (расшифровка символа «Предопределенный процесс») представлены на рис. 1.4, *а* и *б* соответственно.



а *б*

Рис. 1.4

Обозначения схемы алгоритма

$x1$, $y1$ – фактические параметры, обозначающие значения переменных x и y соответственно, используемые для передачи значений в функцию Z;

$Z(x1, y1)$ – вызов функции Z для вычисления значения $z = \sin[\cos^4(x \cdot y)]$;

$z1$ –возвращаемое значение функции Z.

Программа

```
Program Prog1_4;
Var
   z1, x1, y1 : Real;
Function Z(x, y : Real) : Real;
Begin                    {начало функции z}
   z := sin(cos(x*y) * cos(x*y) * cos(x*y) * cos(x*y));
End;                     {конец функции z}
Begin                    {начало основной программы}
   Write('Введите значения x, y ');
   Readln(x1, y1);       {чтение значений переменных x1, y1}
   z1 := Z(x1, y1);      {вычисление z1 с использованием функции z}
   Writeln('z =' , z1:7:2);{вывод на экран}
end.                     {конец основной программы}
```

Результат работы программы

```
Введите значения x, y 1 -2
z =    0.03
```

1.4. **Алгоритмы и программы табуляции функции**

Задача 1.5. Разработать алгоритм и программу для табуляции функции $y = \sin z_i + e^{\alpha t_i}$, где переменные z_i, t_i (i = 1; n) являются массивами. Число элементов массивов n вводить с клавиатуры. Значения y выводить на экран.

Решение задачи

Алгоритм процесса табуляции состоит в последовательном вычислении функции с заданным шагом по *переменному* аргументу. Требуется организация цикла от 1 до n для перебора значений массивов z_i и t_i. Схема алгоритма решения задачи представлена на рис. 1.5.

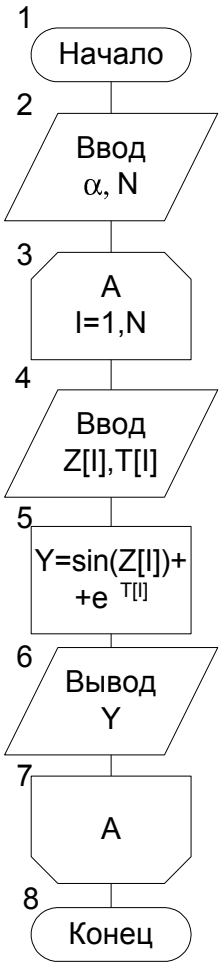


Рис. 1.5

Программа

```
Program Prog1_5;
Var
   N, I : Byte;
   Z, T : array[1..20] of Real;
   Y, a : Real;
Begin              {начало раздела операторов программы}
   Write('Введите число элементов массивов (до 20) ');
   Readln(N);      {чтение значения числа элементов массивов}
   Write('Введите значение коэффициента a ');
   Readln(a);      {чтение значение коэффициента a}
   for I := 1 to N do
   Begin           {начало цикла}
     Write('Введите ', I, '-е элементы массивов Z и T ');
     Readln(Z[I], T[I]);{чтение значений элементов массивов Z, T}
     Y := sin(Z[I]) * Exp(a*T[I]);{вычисление Y}
     Writeln('Y = ', Y:7:2);      {вывод на экран значения Y}
   end;            {конец цикла}
End.               {конец программы}
```

Результат работы программы

```
Введите число элементов массивов (до 20)  3
Введите значение коэффициента a  1.6
Введите 1-е элементы массивов Z и T  2.1  3.2
Y = 144.45
Введите 2-е элементы массивов Z и T  0.2  -1.5
Y = 0.02
Введите 3-е элементы массивов Z и T 1.4  -2.1
Y = 0.03
```

Обозначения схемы алгоритма

$Z[I]$ – z_i элемент массива Z;
$T[I]$ – t_i элемент массива T.

Пошаговый словесный алгоритм

Шаг 1. Ввести значения α, N.

Начало цикла – первая итерация

Шаг 2. Присвоить $I = 1$.

Шаг 3. Ввести значения Z_1, T_1.

Шаг 4. Вычислить значение $Y = \sin(Z_1) + e^{\alpha T_1}$.

Шаг 5. Вывести значение на экран.

Вторая итерация цикла

Шаг 6. Присвоить $I = 2$.

Шаг 7. Ввести значения Z_2, T_2.

Шаг 8. Вычислить значение $Y = \sin(Z_2) + e^{\alpha T_2}$.

Шаг 9. Вывести значение Y на экран.

...

N-я итерация цикла

Шаг 10. Присвоить $I = N$.

Шаг 11. Ввести значения Z_N, T_N.

Шаг 12. Вычислить значение $Y = \sin(Z_N) + e^{\alpha T_N}$.

Шаг 13. Вывести значение Y на экран.

Конец цикла

***Задача** 1.6.* Разработать схему алгоритма и программу табуляции функции с ограниченной областью определения $y = \log_3(x^3 - 4) + \sqrt{\sin x^2}$, если переменная $x = -10; 100$ (0,1). Значения y выводить на экран. Если x не входит в область определения функции, то вывести сообщение о том, что при данном x нет значений.

Решение задачи

Алгоритм решения задачи состоит в вычислении для каждого значения x значений y с учетом области определения, а именно $x^3 - 4 > 0$ и $\sin x^2 \geq 0$. Логика алгоритма требует организации цикла для изменения значений x и использования разветвляющихся процессов для учета области определения функции. Схема алгоритма решения задачи представлена на рис. 1.6 (перебор значений x организован с использованием разветвляющейся структуры).

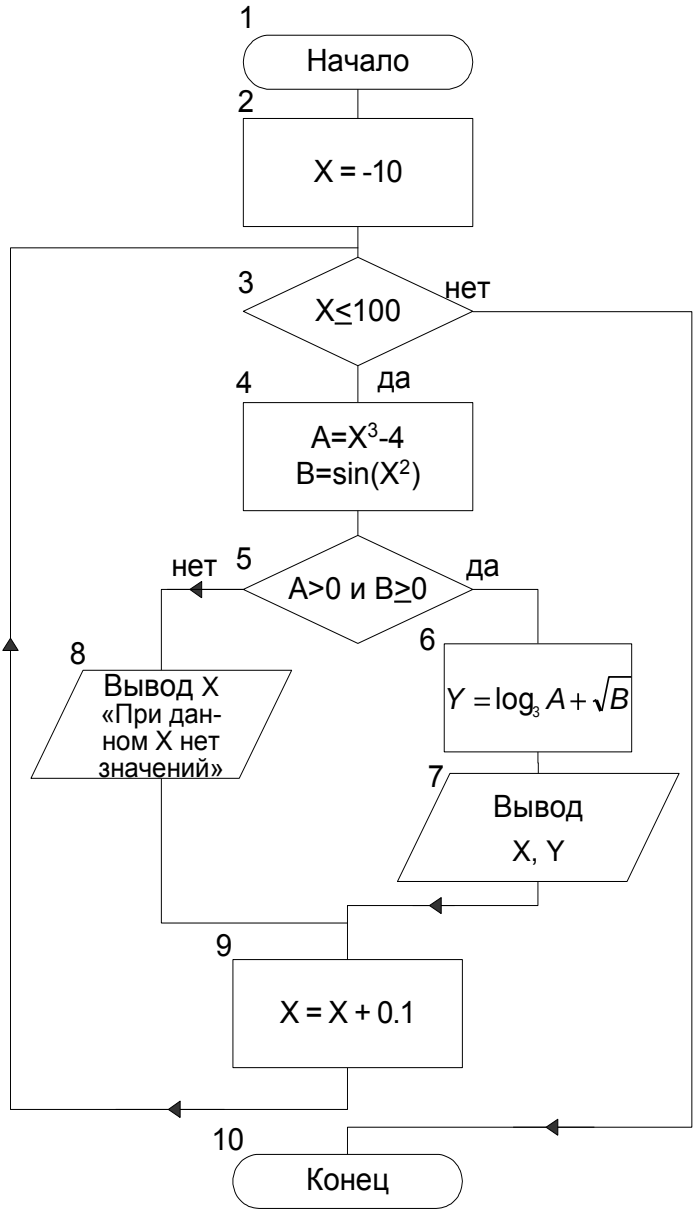

1
Начало
2
X = -10
3
X≤100
нет
да
4
A=X3-4
B=sin(X2)
5
A>0 и B≥0
нет
да
8
Вывод X
«При дан-
ном X нет
значений»
6
Y = log3 A + √B
7
Вывод
X, Y
9
X = X + 0.1
10
Конец

Рис. 1.6

Пояснения к схеме алгоритма

Обозначения:

A – промежуточная переменная для вычисления условия $x^3 - 4$;

B – промежуточная переменная для вычисления условия $\sin x^2$.

Символ 1. Начало алгоритма.

Символ 2. Присвоение переменной *X* первоначального значения –10.

Символ 3. Проверка условия окончания цикла $X \le 100$. Если условие верно, то далее выполняется *символ 4*, если нет, то происходит выход из цикла на *символ 10*.

Символ 4. Вычисление значений промежуточных переменных *A* и *B*.

Символ 5. Проверка условия принадлежности значения *X* области определения функции *Y*. Если условие верно, то далее выполняются *символы 6, 7*, если нет – *символ 8*.

Символ 6. Вычисление значения *Y* .

Символ 7. Вывод на экран значений *X*, *Y*.

Символ 8. Вывод на экран значения *X* и сообщения «При данном *X* нет значений».

Символ 9. Приращение переменной *X* на шаг 0,1.

Символ 10. Конец алгоритма.

Программа

```
Program Prog1_6;
Var
   X, Y, A, B : Real;
Begin                {начало раздела операторов программы}
   X := -10;         {начальное значение переменной X}
  While X <= 100 do {выполнять цикл пока X≤100}
  Begin              {начало цикла}
    A := X*X*X - 4; B := Sin(X*X); {вычисление промежут. значений}
    If (A > 0) and (B >= 0) Then {проверка области определения}
    Begin
      Y := Ln(A)/Ln(3) +Sqrt(B);          {вычисление Y}
      Writeln('X = ', X:5:1, ' Y = ', Y:7:2); {вывод на экран}
    end else Writeln('X = ', X:5:1, ' При данном X нет значений ');
    X := X + 0.1;          {приращение переменной X}
  end;                     {конец цикла}
End.                       {конец программы}
```

Фрагмент результата работы программы

```
X = -10.0  При данном X нет значений
X =  -9.9  При данном X нет значений
X =  -9.8  При данном X нет значений
…
X =  99.7 Y = 12.87
X =  99.8 Y = 13.53
X =  99.9 Y = 13.43
X = 100.0  При данном X нет значений
```

1.5. **Алгоритмы и программы для обработки массивов**

Задача 1.7. Разработать словесный алгоритм, схему алгоритма и программу нахождения максимального значения одномерного массива *A*, состоящего из четырёх элементов: 30, 20, 40 и 50.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 1.7.

Программа

```
Program Prog1_7;
Var
   A : array [1..4] of Byte;
   Max, I : Byte;
Begin                {начало раздела операторов программы}
   {заполнение массива значениями}
   A[1] := 30; A[2] := 20; A[3] := 40; A[4] := 50;
   Max := A[1];      {первоначальное значение переменой максимума}
   for I := 2 to 4 do {цикл для перебора элементов массива A}
       If Max < A[I] Then Max := A[I]; {проверка условия на максимум}
   Writeln('Максимальный элемент массива = ', Max); {вывод на экран}
End.                 {конец программы}
```

Результат работы программы

```
Максимальный элемент массива = 50
```

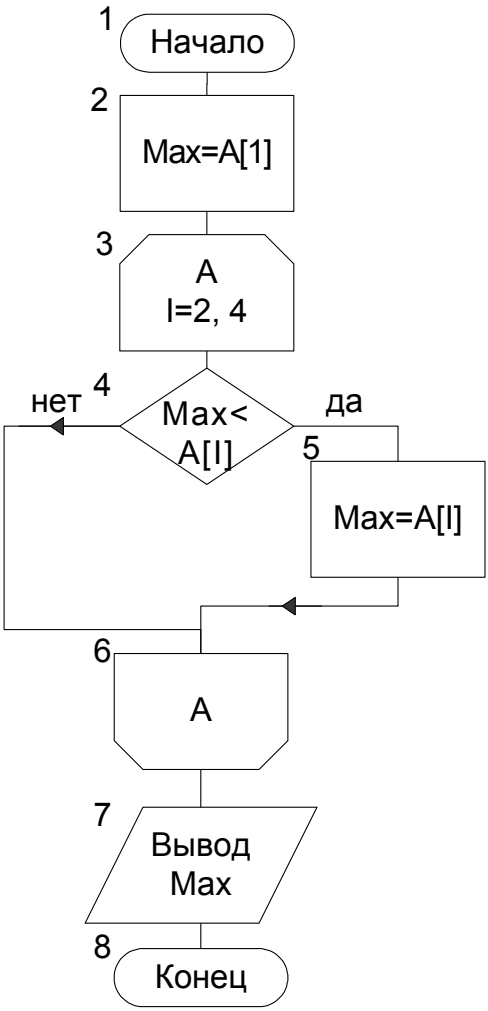


Рис. 1.7

Пояснения к схеме алгоритма

Обозначения:
A[*I*] – a_i элемент массива *A*;
Max – значение максимального элемента массива *A*.

Символ 1. Начало алгоритма.

Символ 2. Присвоение переменной *Max* значения первого элемента массива *A*[1].

Символ 3. Открытие цикла с параметром *I* = 2; 4 для перебора элементов массива *A*.

Символ 4. Проверка условия на максимум. Если переменная *Max* меньше рассматриваемого в текущей итерации элемента массива *A*[*I*], то выполняется *символ 5*, иначе – *символ 6*.

Символ 5. Присвоение переменной *Max* значения *A*[*I*].

Символ 6. Закрытие цикла с параметром *I*.

Символ 7. Вывод на экран значения переменной *Max*.

Символ 8. Конец алгоритма.

Пошаговый словесный алгоритм

Шаг 1. Присвоить переменной *Max* значение *A*[1] (*Max* = 30).

```
Начало цикла
```

Шаг 2. Присвоить *I* = 2 (первая итерация цикла).

Шаг 3. Проверка условия на максимум: 30 < 20 (условие не выполняется).

```
После первой итерации Max = 30.
```

Шаг 4. Присвоить *I* = 3 (вторая итерация цикла).

Шаг 5. Проверка условия на максимум: 30 < 40 (условие выполняется). Присвоить *Max* = 40.

```
После второй итерации Max = 40.
```

Шаг 6. Присвоить *I* = 4 (третья итерация цикла).

Шаг 7. Проверка условия на максимум: 40 < 50 (условие выполняется). Присвоить *Max* = 50.

```
После третьей итерации Max = 50. Конец цикла.
```

***Задача** 1.8.* Разработать схему алгоритма и программу нахождения суммы индексов первого минимального элемента матрицы *A* для элементов a_{ij}, удовлетворяющих условию $i + j > 10$. Элементы

исходной матрицы A размером 10×20 являются вещественными числами и записаны в файле *a.a*.

Решение задачи

Решение задачи состоит в реализации двух циклов для перебора значений матрицы A и сохранении индексов минимального элемента. Следует обратить внимание, что его следует искать среди элементов, удовлетворяющих условию $i + j > 10$ и, значит, для задания начального значения переменной минимального элемента необходимо использовать элемент a_{ij}, удовлетворяющий этому условию, например, $a_{10\ 20}$ (поскольку $10 + 20 > 10$). Схема алгоритма решения задачи представлена на рис. 1.8.

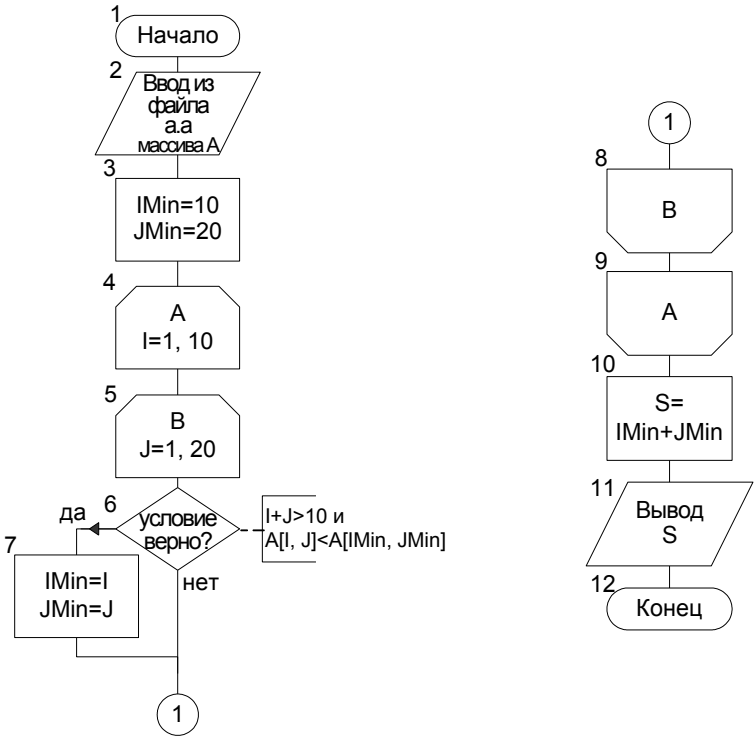


Рис. 1.8

Следует отметить, что схема алгоритма решения задачи организована без поиска минимального элемента матрицы A, так как в условии задачи не требуется его нахождение. Ниже приведен текст программы, составленной по этой схеме.

Программа

```
Program Prog1_8;
Var
   F : Text;
   A : array [1..10, 1..20] of Real;
   I, J, IMin, JMin : Byte;
   S : Real;
Begin          {начало раздела операторов программы}
   Assign(F, 'a.a'); Reset(F);    {открытие файла a.a}
   for I := 1 to 10 do
   Begin       {начало цикла 1 для перебора строк матрицы A}
      for J := 1 to 20 do {цикл 2 для перебора столбцов матрицы A}
        Read(F, A[I, J]); {чтение элемента A[I, J] из файла a.a}
      Readln(F);          {переход на след. строку в файле a.a}
   end;                   {конец цикла 1}
   Close(F);              {закрытие файла a.a}
   {задание начальных значений для индексов минимального элемента}
   IMin := 10;  JMin := 20;
   for I :=1 to 10 do   {цикл 3 для перебора строк матрицы A}
      for J:= 1 to 20 do {цикл 4 для перебора столбцов матрицы A}
         {проверка условий}
         If (I+J>10) and (A[I,J]<A[IMin,JMin]) Then
         Begin          {начало блока 1}
            {сохранение значений индексов меньшего элемента}
            IMin := I; JMin := J;
         end;           {конец блока 1, конец циклов 3, 4}
   S := IMin + JMin;              {вычисление суммы индексов}
   Writeln('S = ', S :7:2);       {вывод суммы на экран}
End.                              {конец программы}
```

Пояснения к схеме алгоритма

Обозначения:
$A[I, J] – a_{ij}$ элемент матрицы A;

IMin, *JMin* – значения индексов строки и столбца минимального элемента матрицы *A* среди элементов a_{ij}, удовлетворяющих условию $i + j > 10$;

S – сумма индексов строк и столбцов минимального элемента матрицы *A* среди элементов a_{ij}, удовлетворяющих условию $i + j > 10$.

Символ 1. Начало алгоритма.

Символ 2. Ввод двумерного массива из файла *a.a*. Ввод значений производится с организацией внешнего цикла для перебора строк матрицы *A* и внутреннего цикла для перебора столбцов матрицы (табл. П 1.13), на схеме для упрощения это показано в одном символе.

Символ 3. Присвоение переменным *IMin*, *JMin* значений 10 и 20 (первоначально элемент должен удовлетворять условию $I + J > 10$).

Символ 4. Открытие внешнего цикла с параметром $I = 1, 10$ для перебора строк матрицы *A*.

Символ 5. Открытие внутреннего цикла с параметром $J = 1, 20$ для перебора столбцов матрицы *A*.

Символ 6. Проверка условия на минимум и условия $I + J > 10$. Если условия верны, то выполняется *символ 7*, иначе – *символ 8*.

Символ 7. Присвоение переменной *IMin* значения номера строки *I*, переменной *JMin* – значения номера столбца *J*.

Символ 8. Закрытие внутреннего цикла с параметром *J*.

Символ 9. Закрытие внешнего цикла с параметром *I*.

Символ 10. Вычисление суммы индексов (переменная *S*) строк и столбцов минимального элемента.

Символ 11. Вывод на экран значения переменной *S*.

Символ 12. Конец алгоритма.

***Задача** 1.9.* Разработать схему алгоритма и программу для вычисления значений интервальной функции *z* для значений элементов массива x_i ($i = 1; n$), которые заданы с клавиатуры:

$$z = \begin{cases} \sin x_i, & \text{если } x_i < -10; \\ \cos x_i, & \text{если } -10 \le x_i < 1; \\ \ln x_i, & \text{если } 1 \le x_i < 10; \\ \operatorname{tg} x_i, & \text{если } x_i \ge 10. \end{cases}$$

Значения *z* выводятся на экран.

Решение задачи

Алгоритм решения задачи состоит в организации цикла для перебора элементов массива X (проводится последовательная проверка заданных интервальных условий переменной x_i и вычисление z в соответствии с постановкой задачи). Обратить внимание, что тангенс должен быть определен, то есть $\cos x_i \neq 0$. Схема алгоритма решения задачи представлена на рис. 1.9.

Программа

```
Program Prog1_9;
Var
   N, I : Byte;
   X : Array [1..20] of Real;
   Z: Real;
Begin          {начало раздела операторов программы}
   Write('Введите размер (число элементов) массива X (до 20) ');
   Read(N);   {чтение числа элементов массива X}
   for I := 1 to N do
   Begin  {начало цикла-для перебора элементов массива X}
    Write('Введите ', I, '-й элемент массива X  ');
    Readln(X[I]); {чтение значения элемента X[I]}
    {вычисление Z в зависимости от значения элемента X[I]}
    If (X[I]>=10) and (Cos(X[I])=0)
      Then Writeln('Функция не определена')
      else begin           {начало блока 1}
        If X[I]<-10 Then Z := Sin(X[I])
          else If X[I] < 1 Then Z := cos(X[I])
            else If X[I]<10 Then Z:= Ln(X[I])
              else Z:=sin(X[I])/cos(X[I]);
               Writeln('Z = ', Z:7:2); {вывод значения Z на экран}
         end;              {конец блока 1}
   end;                    {конец цикла}
End.                       {конец программы}
```

Результат работы программы

```
Введите размер (число элементов) массива X (до 20) 3
Введите 1-й элемент массива X  6
Z =    1.79
Введите 2-й элемент массива X  30
Z =   -6.41
Введите 3-й элемент массива X  -10
Z =   -0.84
```

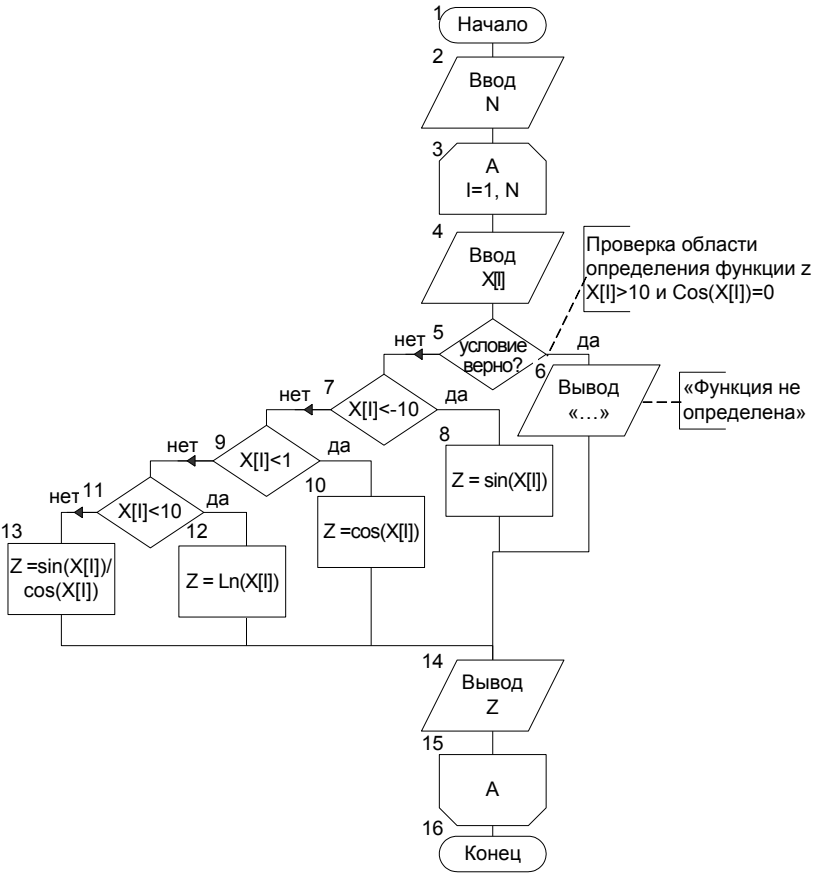


Рис. 1.9

Пояснения к схеме алгоритма

Обозначение:

$X[I] - x_i$ элемент массива X.

Символ 1. Начало алгоритма.

Символ 2. Ввод размера массива N.

Символ 3. Открытие цикла с параметром $I = 1$; N для перебора значений массива $X[I]$.

Символ 4. Ввод с клавиатуры элемента массива $X[I]$.

Символ 5. Проверка условия $X[I] \geq 10$ и $\cos(X[I]) = 0$ (условие того, что значение $\operatorname{tg}(X[I]) = \dfrac{\sin(X[I])}{\cos(X[I])}$ может быть вычислено). Если условие верно, то далее выполняется *символ 6*, если нет – *символ 7*.

Символ 6. Вывод на экран сообщения «Функция не определена».

Символ 7. Проверка условия $X[I] < -10$. Если условие верно, то далее выполняется *символ 8*, если нет – *символ 9*.

Символ 8. Присвоение переменной Z значения $\sin(X[I])$.

Символ 9. Проверка условия принадлежности $X[I]$ интервалу $[-10; 1)$. Если условие верно, то далее выполняется *символ 10*, если нет – *символ 11*.

Символ 10. Присвоение переменной Z значения $\cos(X[I])$.

Символ 11. Проверка условия принадлежности $X[I]$ интервалу $[1; 10)$. Если условие верно, то далее выполняется *символ 12*, если нет – *символ 13*.

Символ 12. Присвоение переменной Z значения $\ln(X[I])$.

Символ 13. Присвоение переменной Z значения $\dfrac{\sin(X[I])}{\cos(X[I])}$.

Символ 14. Вывод на экран значения переменной Z.

Символ 15. Закрытие цикла с параметром I.

Символ 16. Конец алгоритма.

***Задача** 1.10.* Разработать схему алгоритма и программу для вычисления значений элементов матрицы A размером 5×12, которые заданы формулой

$$a_{ij} = \begin{cases} \sin(i + j), \text{если } i \leq j; \\ \ln(\dfrac{i}{j}), \text{если } i > j. \end{cases}$$

Значения элементов массива a_{ij} вывести на экран.

Решение задачи

Алгоритм решения задачи состоит в организации двух циклов по i и j для вычисления в зависимости от условий значений элементов

двумерного массива a_{ij}. Схема алгоритма решения задачи представлена на рис. 1.10.

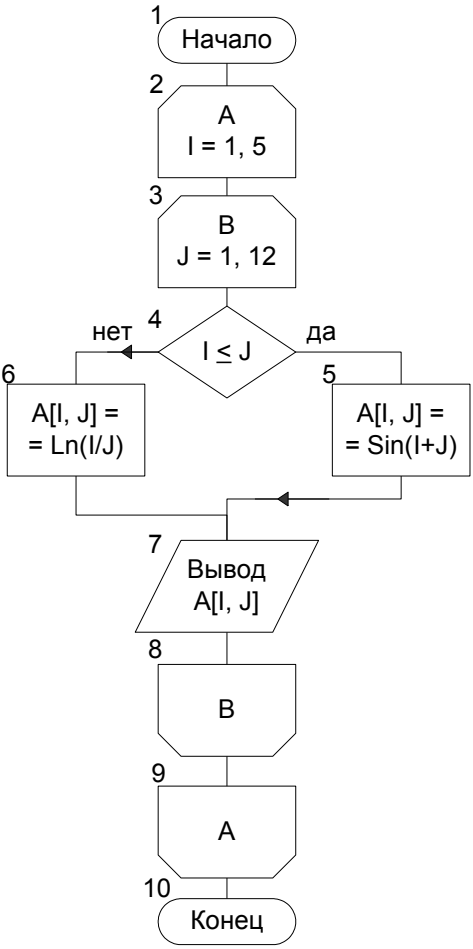


Рис. 1.10

Пояснения к схеме алгоритма

Обозначения:
$A[I, J] – a_{ij}$ элемент матрицы A.

Символ 1. Начало алгоритма.

Символ 2. Открытие внешнего цикла с параметром $I = 1; 5$ для перебора строк матрицы A.

Символ 3. Открытие внутреннего цикла с параметром $J = 1; 12$ для перебора столбцов матрицы A.

Символ 4. Проверка условия $I \le J$. Если условие верно, то далее выполняется *символ 5*, если нет – то *символ 6*.

Символ 5. Присвоение элементу $A[I, J]$ значения $\sin(I + J)$.

Символ 6. Присвоение элементу $A[I, J]$ значения $\ln(I/J)$.

Символ 7. Вывод на экран значения $A[I, J]$.

Символ 8. Закрытие внутреннего цикла с параметром J.

Символ 9. Закрытие внешнего цикла с параметром I.

Символ 10. Конец алгоритма.

Программа

```
Program Prog1_10;
Var
   I, J : Byte;
   A : array [1..5, 1..12] of Real;
Begin          {начало раздела операторов программы}
   for I:= 1 to 5 do
   Begin       {начало цикла 1 - для перебора строк массива A}
     for J := 1 to 12 do
     Begin  {начало цикла 2 - для перебора столбцов массива A}
       {вычисление значения элемента массива в зависимости от условия}
       If I <= J Then A[I, J] := sin(I + J)
                 else A[I, J] := ln(I/J);
       Write(A[I, J]:7:2, ' ');{вывод элемента массива на экран}
     end;      {конец цикла 2}
     Writeln; {переход на следующую строку}
   end;        {конец цикла 1}
End.           {конец программы}
```

Результат работы программы

```
0.91  0.14  -0.76  -0.96  -0.28  0.66  0.99  0.41 -0.54 -1.00 -0.54  0.42
0.69 -0.76  -0.96  -0.28   0.66  0.99  0.41 -0.54 -1.00 -0.54  0.42  0.99
1.10  0.41  -0.28   0.66   0.99  0.41 -0.54 -1.00 -0.54  0.42  0.99  0.65
1.39  0.69   0.29   0.99   0.41 -0.54 -1.00 -0.54  0.42  0.99  0.65 -0.29
1.61  0.92   0.51   0.22  -0.54 -1.00 -0.54  0.42  0.99  0.65 -0.29 -0.96
```

Задача 1.11. Разработать схему алгоритма и программу получения из матрицы $A_{10\times10}$, заданной по закону $a_{ij} = \cos(i + j)$, двух одномерных массивов C и D. В массив C занести все неотрицательные элементы заштрихованной области матрицы A, а в массив D – отрицательные элементы незаштрихованной области. Вывести на экран число элементов массивов C и D.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} & a_{19} & a_{1\,10} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} & a_{28} & a_{29} & a_{2\,10} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} & a_{38} & a_{39} & a_{3\,10} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} & a_{49} & a_{4\,10} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} & a_{58} & a_{59} & a_{5\,10} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} & a_{68} & a_{69} & a_{6\,10} \\ a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} & a_{78} & a_{79} & a_{7\,10} \\ a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} & a_{87} & a_{88} & a_{89} & a_{8\,10} \\ a_{91} & a_{92} & a_{93} & a_{94} & a_{95} & a_{96} & a_{97} & a_{98} & a_{99} & a_{9\,10} \\ a_{10\,1} & a_{10\,2} & a_{10\,3} & a_{10\,4} & a_{10\,5} & a_{10\,6} & a_{10\,7} & a_{10\,8} & a_{10\,9} & a_{10\,10} \end{bmatrix}$$

Решение задачи

Алгоритм решения задачи включает анализ двух условий:

$$\begin{cases} j \ge i, \\ a_{ij} \ge 0 \end{cases} \text{ и } \begin{cases} j < i, \\ a_{ij} < 0. \end{cases}$$

Если выполняется первое условие, то элемент a_{ij} заносится в массив C: $c_k = a_{ij}$, если выполняется второе условие – то элемент a_{ij} заносится в массив D: $d_n = a_{ij}$. Таким образом, логика алгоритма требует организации двух циклов для формирования и перебора значений a_{ij} и использования разветвляющихся процессов для их анализа. Схема алгоритма решения задачи представлена на рис. 1.11.

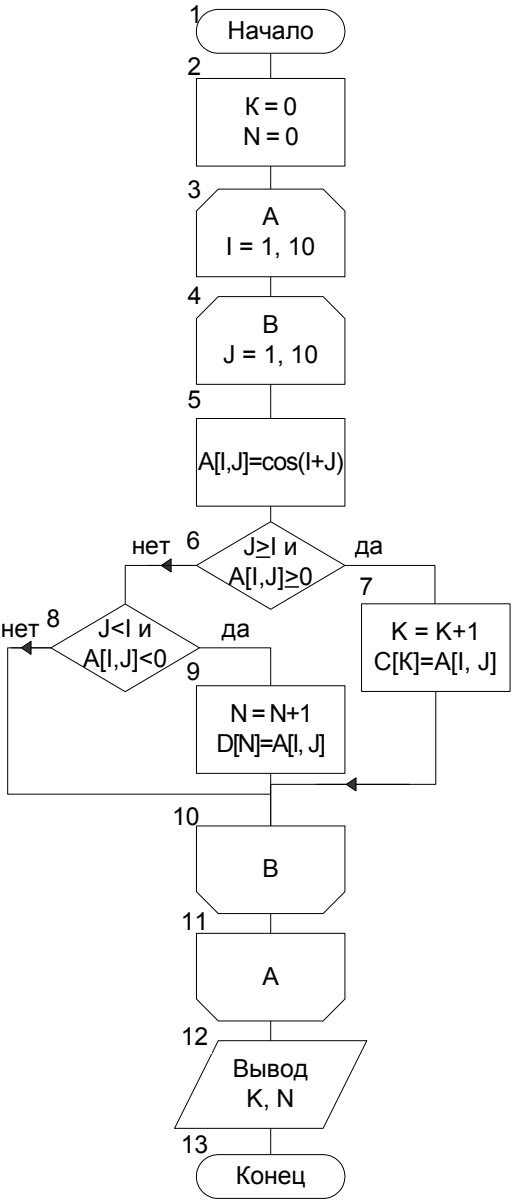

1
Начало
2
K = 0
N = 0
3
A
I = 1, 10
4
B
J = 1, 10
5
A[I,J]=cos(I+J)
нет
6
J≥I и
A[I,J]≥0
да
7
K = K+1
C[K]=A[I, J]
нет
8
J<I и
A[I,J]<0
да
9
N = N+1
D[N]=A[I, J]
10
B
11
A
12
Вывод
K, N
13
Конец

Рис. 1.11

Пояснения к схеме алгоритма

Обозначения:

$A[I, J] – a_{ij}$ элемент матрицы A;

$C[K] – c_k$ элемент массива C;

$D[N] – d_n$ элемент массива D;

K, N – счетчики числа элементов массивов C, D.

Символ 1. Начало алгоритма.

Символ 2. Обнуление счетчиков элементов массивов C (переменная K) и массива D (переменная N).

Символ 3. Открытие внешнего цикла с параметром $I = 1; 10$ для перебора строк матрицы A.

Символ 4. Открытие внутреннего цикла с параметром $J = 1; 10$ для перебора столбцов матрицы A.

Символ 5. Вычисление элемента $A[I, J]$.

Символ 6. Проверка условий принадлежности элемента $A[I, J]$ заштрихованной области ($J \geq I$) и неотрицательности элемента $A[I, J]$. Если условия верны, выполняется *символ 7*, если нет – *символ 8*.

Символ 7. Увеличение счетчика элементов массива C (переменная K). Присвоение элементу массива C значения элемента $A[I, J]$.

Символ 8. Проверка условий принадлежности элемента $A[I, J]$ незаштрихованной области ($J < I$) и отрицательности элемента $A[I, J]$. Если условия верны, выполняется *символ 9*, если нет – *символ 10*.

Символ 9. Увеличение счетчика элементов массива D (переменная N). Присвоение элементу массива D значения элемента $A[I, J]$.

Символ 10. Закрытие внутреннего цикла с параметром J.

Символ 11. Закрытие внешнего цикла с параметром I.

Символ 12. Вывод на экран значений K, N.

Символ 13. Конец алгоритма.

Программа

```
Program Prog1_11;
Var
   I, J, K, N : Byte;
   A : array [1..10, 1..10] of Real;
   C, D : array [1..55] of Real;
Begin            {начало раздела операторов программы}
   K := 0; N := 0; {начальные значения для индексов массивов C, D}
```

```
   for I := 1 to 10 do {цикл 1 – для перебора строк матрицы A}
     for J := 1 to 10 do {цикл 2 – для перебора столбцов матрицы A}
     Begin              {начало циклов 1, 2}
       A[I, J] := cos(I + J);     {вычисление A[I, J]}
       If (J >= I) and (A[I, J] >= 0) Then {проверка условий}
       Begin          {начало блока 1 – условия верны}
         K := K + 1;        {увеличение индекса K на 1}
         C[K] := A[I, J];  {заполнение массива C}
       end                {конец блока 1}
       else If (J < I) and (A[I, J] < 0) Then {проверка условий}
            Begin         {начало блока 2-условия верны}
              N := N +  1;      {увеличение индекса N на 1}
              D[N] := A[I, J]; {заполнение массива D}
            end;          {конец блока 2}
     end;                 {конец циклов 1, 2}
   Writeln('Число элементов массива C = ', K);  {вывод K}
   Writeln('Число элементов массива D = ', N); {вывод N}
End.          {конец программы}
```

Результат работы программы

```
Число элементов массива C = 30
Число элементов массива D = 20
```

1.6. **Алгоритмы и программы для нахождения сумм и произведений функциональных выражений**

***Задача** 1.12.* Разработать схему алгоритма и написать программу для вычисления значения $A = \dfrac{\prod_{k=1}^{20} \sin(k+1)}{\sum_{i=1}^{100} \cos(i)}$. Значение A вывести на экран.

Решение задачи

Алгоритм решения задачи сводится к последовательному вычислению знаменателя (суммирование), при этом следует обратить внимание на то, что он не должен быть равен нулю, а затем – числителя (произведение). Схема алгоритма решения задачи представлена на рис. 1.12.

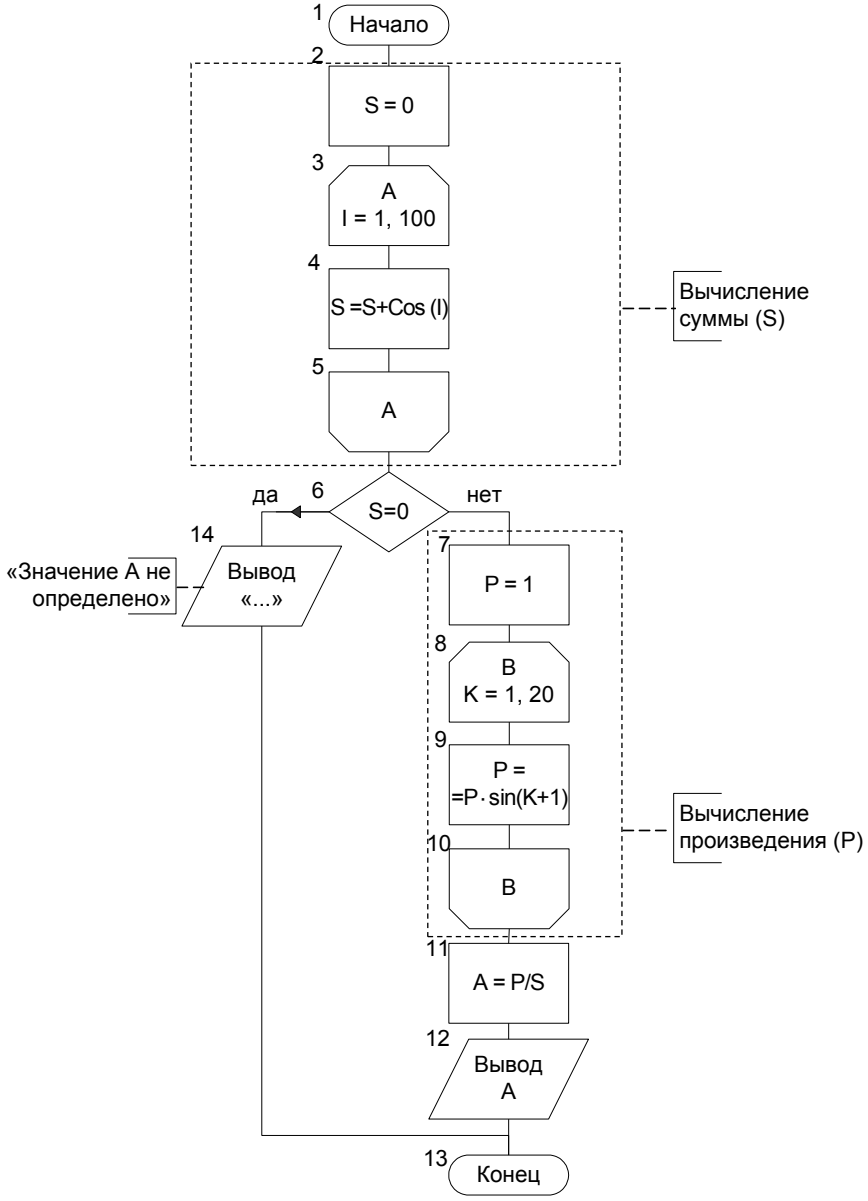

1
Начало
2
S = 0
3
A
I = 1, 100
4
S =S+Cos (I)
5
A
Вычисление суммы (S)
6
S=0
да
нет
14
«Значение A не определено»
Вывод «...»
7
P = 1
8
B
K = 1, 20
9
P = =P·sin(K+1)
10
B
Вычисление произведения (P)
11
A = P/S
12
Вывод A
13
Конец

Рис. 1.12

Пояснения к схеме алгоритма

Обозначения:

P – значение выражения произведения $\prod_{k=1}^{20} \sin(k+1)$;

S – значение выражения суммы $\sum_{i=1}^{100} \cos(i)$.

Символ 1. Начало алгоритма.

Символ 2. Присвоение переменной *S* начального значения 0.

Символ 3. Открытие цикла с параметром *I* = 1; 100 для вычисления суммы.

Символ 4. Вычисление переменной *S* по формуле с учетом накоплений сумм.

Символ 5. Закрытие цикла с параметром *I*.

Символ 6. Проверка условия равенства нулю значения *S*. Если верно, то выполняется *символ 14*, если нет – *символ 7*.

Символ 7. Присвоение переменной *P* начального значения 1.

Символ 8. Открытие цикла с параметром *K* = 1; 20 для вычисления произведения.

Символ 9. Вычисление переменной *P* по формуле с учетом накоплений произведений.

Символ 10. Закрытие цикла с параметром *K*.

Символ 11. Вычисление переменной *A*.

Символ 12. Вывод на экран значения *A*.

Символ 13. Конец алгоритма.

Символ 14. Вывод на экран сообщения «Значение *A* не определено».

Программа

```
Program Prog1_12;
Var
   K, I : Byte;
   P, S, A : Real;
Begin                {начало раздела операторов программы}
   S := 0;                  {начальное значение переменной S}
   for I := 1 to 100 do
      S := S + cos(I);      {вычисление суммы в цикле}
   If S = 0 Then Writeln('Значение A не определено')
```

```
   else begin                {начало блока 1}
          P := 1;            {начальное значение переменной P}
          for K := 1 to 20 do
            P:=P * sin(K+1);{вычисление произведения в цикле}
          A := P/S;          {вычисление A}
          Writeln('A = ', A:15:12){вывод на экран значения A}
        end;                 {конец блока 1}
End.                         {конец программы}
```

Результат работы программы

```
A=0.00002461930
```

1.7. **Алгоритм и программа метода пузырьковой сортировки**

Задача 1.13. Разработать схему алгоритма и написать программу сортировки по возрастанию одномерного массива *A*, состоящего из пяти элементов: 10, 80, 45, 36, 71. Использовать метод пузырьковой сортировки. Значения элементов отсортированного массива вывести на экран.

Решение задачи

Алгоритм пузырьковой сортировки заключается в повторяющихся проходах по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно, и, если порядок в паре неверный, их меняют местами. При первом проходе сравниваются первый и второй элементы, затем второй и третий и т. д., пока не будет достигнут последний элемент массива. Таким образом, наибольший элемент оказывается в конце массива, то есть «всплывает», как пузырёк в воде. При втором проходе производятся аналогичные сравнения, которые останавливаются на предпоследнем элементе массива, и, таким образом, после второго прохода на предпоследнем месте окажется второй наибольший элемент и т. д.

В нашем случае, поскольку массив состоит из пяти элементов, будет сделано четыре прохода, и на каждом проходе все элементы будут попарно сравниваться. Таким образом, при построении схемы алгоритма необходимо организовать два цикла *A* и *B*. Цикл *B* вводится для попарного сравнения элементов, цикл *A* – для задания числа проходов и номера последнего элемента для сравнения. Схема алгоритма решения задачи представлена на рис. 1.13.

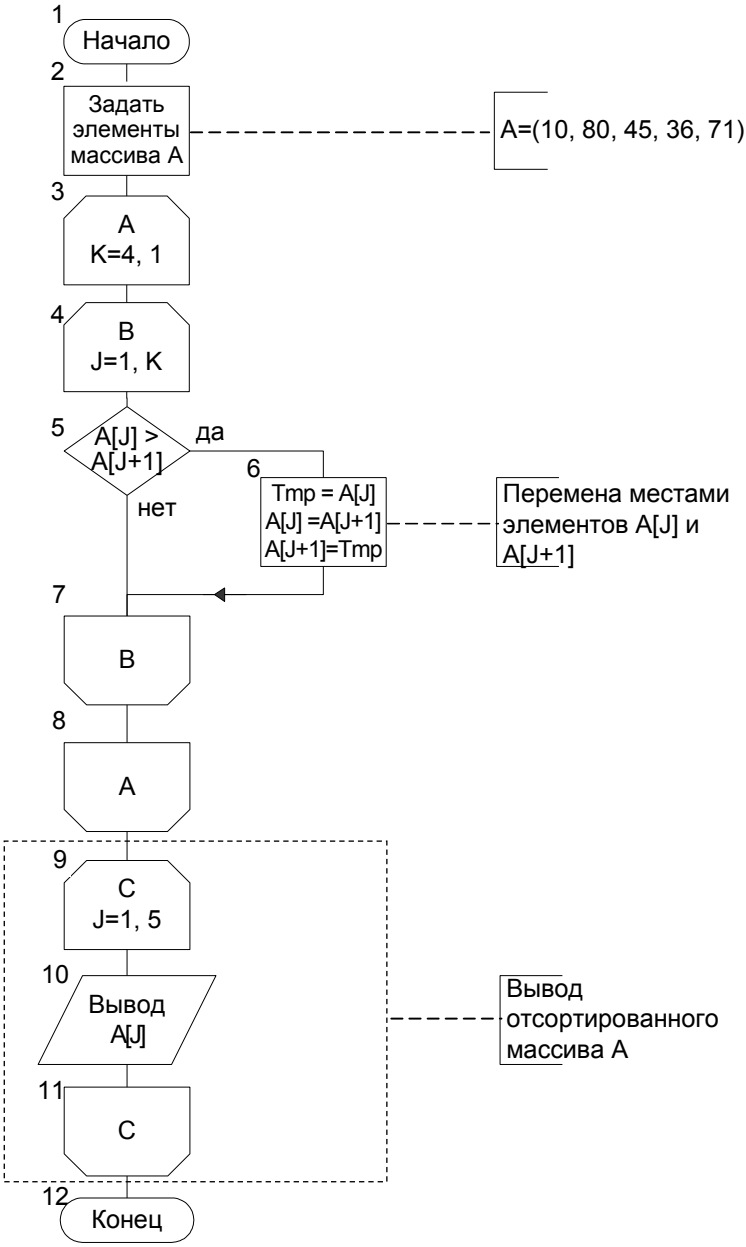

1
Начало
2
Задать элементы массива A
A=(10, 80, 45, 36, 71)
3
A
K=4, 1
4
B
J=1, K
5
A[J] > A[J+1]
да
нет
6
Tmp = A[J]
A[J] =A[J+1]
A[J+1]=Tmp
Перемена местами элементов A[J] и A[J+1]
7
B
8
A
9
C
J=1, 5
10
Вывод A[J]
11
C
Вывод отсортированного массива A
12
Конец

Рис. 1.13

Программа

```
Program Prog1_13;
Var
   A : array [1..5] of Byte;
   K, J, Tmp : Byte;
Begin                {начало раздела операторов программы}
   {заполнение массива A значениями}
   A[1] := 10; A[2] := 80; A[3] := 45; A[4] := 36; A[5] := 71;
   Write('Исходный массив');
   for J := 1 to 5 do Write(A[J], ' '); {вывод исходного массива A}
   Writeln;
   {организация циклов 1, 2 для сравнения элементов массива A}
   for K := 4 downto 1 do
       for J := 1 to K do
          {проверка, если больший элемент находится перед меньшим}
          If A[J] > A[J + 1] Then
          Begin      {начало блока 1}
            {перестановка местами элементов A[J] и A[J+1]}
            Tmp := A[J];   A[J] := A[J+1];   A[J+1] := Tmp;
          end;       {конец блока 1, конец циклов 1, 2}
   Write(Отсортированный массив  ');
   for J := 1 to 5 do  Write(A[J], ' '); {вывод отсортиров. массива A}
End.                 {конец программы}
```

Результат работы программы

```
Исходный массив  10 80 45 36 71
Отсортированный массив  10 36 45 71 80
```

Пояснения к схеме алгоритма

Обозначения

$A[J]$, $A[J + 1]$ – a_j, a_{j+1} элементы массива A;

Tmp – переменная для временного хранения значения a_j.

Символ 1. Начало алгоритма.

Символ 2. Заполнение массива A значениями.

Символ 3. Открытие цикла с параметром $K = 4; 1$ для перебора элементов массива A.

Символ 4. Открытие цикла с параметром $J = 1; K$ для перебора элементов массива A.

Символ 5. Проверка условия *A*[*J*] > *A*[*J* + 1]. Если условие верно, то выполняется *символ 6*, если нет – *символ 7*.

Символ 6. Перестановка местами элементов массива *A*[*J*] и *A*[*J* + 1] с помощью промежуточной переменной *Tmp*.

Символ 7. Конец цикла с параметром *J*.

Символ 8. Конец цикла с параметром *K*.

Символы 9–11. Вывод на экран значений отсортированного массива *A*.

Символ 12. Конец алгоритма.

Пошаговый алгоритм вычислений

```
Начало цикла A (K=4).
Расположение элементов массива A: 10, 80, 45, 36, 71.
```

Шаг 1. Цикл *B* (*J* = 1). Сравнение элементов *A*[1] = 10 и *A*[2] = 80. Условие *A*[1] > *A*[2] не выполняется, массив *A* остался без изменений.

Шаг 2. Цикл *B* (*J* = 2). Сравнение элементов *A*[2] = 80 и *A*[3] = 45. Условие *A*[2] > *A*[3] выполняется. Элементы *A*[2] и *A*[3] меняются местами.

```
Расположение элементов массива A: 10, 45, 80, 36, 71.
```

Шаг 3. Цикл *B* (*J* = 3). Сравнение элементов *A*[3] = 80 и *A*[4] = 36. Условие *A*[3] > *A*[4] выполняется. Элементы *A*[3] и *A*[4] меняются местами.

```
Расположение элементов массива A:= 10, 45, 36, 80, 71.
```

Шаг 4. Цикл *B* (*J* = 4). Сравнение элементов *A*[4] = 80 и *A*[5] = 71. Условие *A*[4] > *A*[5] выполняется. Элементы *A*[4] и *A*[5] меняются местами.

```
Цикл A (K=3)
Расположение элементов массива A: 10, 45, 36, 71, 80.
```

Шаг 5. Цикл *B* (*J* = 1). Сравнение элементов *A*[1] = 10 и *A*[2] = 45. Условие *A*[1] > *A*[2] не выполняется, массив *A* остался без изменений.

Шаг 6. Цикл *B* (*J* = 2). Сравнение элементов *A*[2] = 45 и *A*[3] = 36. Условие *A*[2] > *A*[3] выполняется. Элементы *A*[2] и *A*[3] меняются местами.

```
Расположение элементов массива A: 10, 36, 45, 71, 80.
```

Шаг 7. Цикл *B* (*J* = 3). Сравнение элементов *A*[3] = 45 и *A*[4] = 71. Условие *A*[3] > *A*[4] не выполняется, массив *A* остается без изменений.

```
Цикл A (K=2).
```

Шаг 8. Цикл *B* (*J* = 1). Сравнение элементов *A*[1] = 10 и *A*[2] = 36. Условие *A*[1] > *A*[2] не выполняется, массив *A* остался без изменений.

Шаг 9. Цикл *B* (*J* = 2). Сравнение элементов *A*[2] = 36 и *A*[3] = 45. Условие *A*[2] > *A*[3] не выполняется массив *A* остался без изменений.

```
Цикл A (K=1).
```

Шаг 10. Цикл *B* (*J* = 1). Сравнение элементов *A*[1] = 10 и *A*[2] = 36. Условие *A*[1] > *A*[2] не выполняется, массив *A* остался без изменений.

```
Конец цикла A.
Расположение элементов массива A: 10, 36, 45, 71, 80.
```

1.8. **Алгоритмы и программы с использованием элементов теории множеств**

Задача 1.14. Разработать схему алгоритма и программу вычисления выражения $S = \dfrac{\max\{a_i\} + \max\{a_j\}}{\min\{a_k\} + \min\{a_l\}}$, где элементы подмножеств вычисляются как $a_m = \sin(m)$, $m = 1; 100$, а индексы элементов подмножеств как $i = 1; 40$, $j = 15; 50$, $k = 40; 70$, $l = 60; 100$. Значение S вывести на экран.

Решение задачи

Обозначим области изменения элементов подмножеств, в которых определяются максимумы и минимумы, как 1 (a_i), 2 (a_j), 3 (a_k), 4 (a_l). Тогда алгоритм решения задачи состоит в организации четырёх циклов по *i, j, k, l* для вычисления, в зависимости от условий, максимального или минимального элемента соответствующих областей массива *A*. Схема алгоритма решения задачи представлена на рис. 1.14.

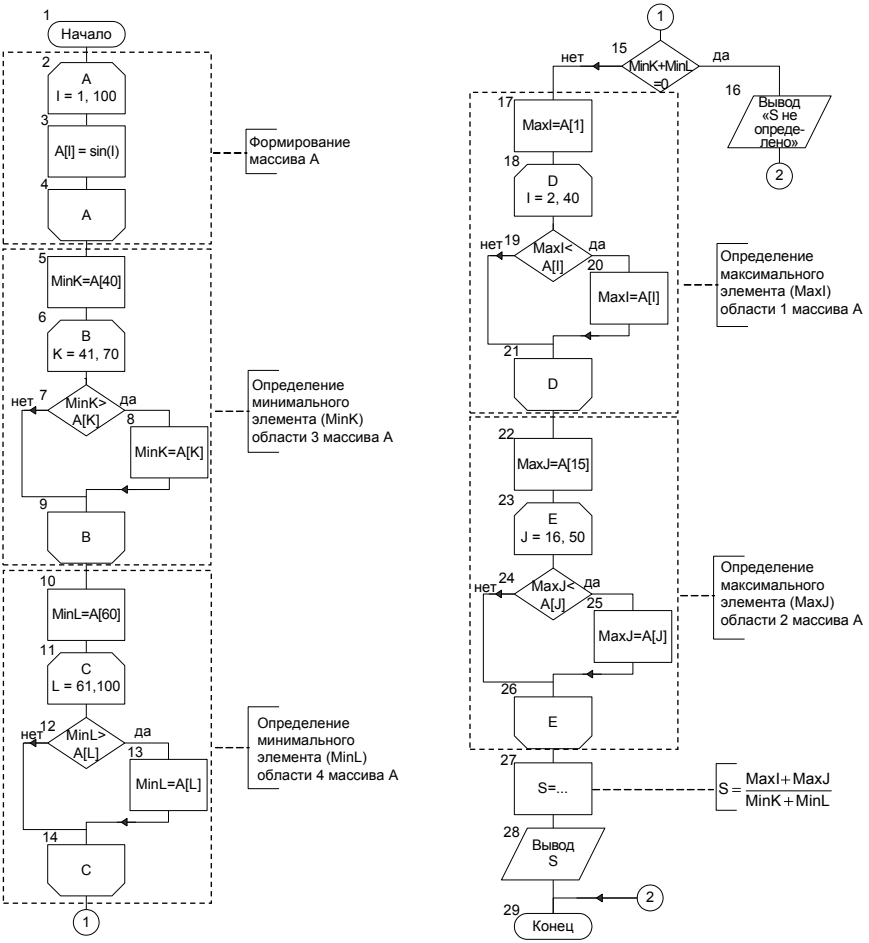


Рис. 1.14

Пояснения к схеме алгоритма

Обозначения:

A[*I*], *A*[*J*], *A*[*K*], *A*[*L*] – a_i, a_j, a_k, a_l элементы массива *A*;

MaxI, *MaxJ* – максимальные элементы областей 1 и 2;

MinK, *MinL* – минимальные элементы областей 3 и 4.

Символ 1. Начало алгоритма.

Символ 2. Открытие цикла с параметром *I* = 1; 100 для перебора элементов массива *A*.

Символ 3. Вычисление *I*-го элемента массива *A*.

Символ 4. Конец цикла с параметром *I*.

Символ 5. Присвоение переменной *MinK* начального значения *A*[40].

Символ 6. Открытие цикла с параметром *K* = 41; 70 для перебора элементов массива *A*.

Символ 7. Проверка условия на минимум. Если переменная *MinK* больше рассматриваемого в текущей итерации элемента массива *A*[*K*], то выполняется *символ 8*, если нет – *символ 9*.

Символ 8. Присвоение переменной *MinK* значения *A*[*K*].

Символ 9. Закрытие цикла с параметром *K*.

Символ 10. Присвоение переменной *MinL* начального значения *A*[60].

Символ 11. Открытие цикла с параметром *L* = 61; 100 для перебора элементов массива *A*.

Символ 12. Проверка условия на минимум. Если переменная *MinL* больше рассматриваемого в текущей итерации элемента массива *A*[*L*], то выполняется *символ 13*, если нет – *символ 14*.

Символ 13. Присвоение переменной *MinL* значения *A*[*L*].

Символ 14. Закрытие цикла с параметром *L*.

Символ 15. Проверка равенства выражения (*MinK* + *MinL*) нулю. Если верно, то выполняется *символ 16*, если нет – *символ 17*.

Символ 16. Вывод на экран сообщения «*S* не определено».

Символ 17. Присвоение переменной *MaxI* начального значения *A*[1].

Символ 18. Открытие цикла с параметром *I* = 2; 40 для перебора элементов массива *A*.

Символ 19. Проверка условия на максимум. Если переменная *MaxI* меньше рассматриваемого в текущей итерации элемента массива *A*[*I*], то выполняется *символ 20*, иначе – *символ 21*.

Символ 20. Присвоение переменной *MaxI* значения *A*[*I*].

Символ 21. Закрытие цикла с параметром *I*.

Символ 22. Присвоение переменной *MaxJ* начального значения *A*[15].

Символ 23. Открытие цикла с параметром *J* = 16; 50 для перебора элементов массива *A*.

Символ 24. Проверка условия на максимум. Если переменная *MaxJ* меньше рассматриваемого в текущей итерации элемента массива *A*[*J*]), то выполняется *символ 25*, если нет – *символ 26*.

Символ 25. Присвоение переменной *MaxJ* значения *A*[*J*].

Символ 26. Закрытие цикла с параметром *J*.

Символ 27. Вычисление значения *S*.

Символ 28. Вывод на экран значения *S*.

Символ 29. Конец алгоритма.

Программа

```
Program Prog1_14;
Var
   I, J, K, L : Byte;
   A : array [1..100] of Real;
   MaxI, MaxJ, MinK, MinL, S : Real;
Begin                    {начало раздела операторов программы}
   for I := 1 to 100 do  {цикл 1 – для перебора элементов массива A}
     A[I] := sin(I);     {вычисление A[I]}
   MinK := A[40];        {первоначальное значение переменной MinK}
   for K := 41 to 70 do {цикл 4 – для перебора элементов области 3}
     if MinK > A[K] then MinK := A[K]; {проверка условия на минимум}
   MinL := A[60];        {первоначальное значение переменной MinL}
   for L := 61 to 100 do {цикл 5 – для перебора элементов области 4}
     If MinL > A[L] then MinL := A[L]; {проверка условия на минимум}
   If (MinK + MinL)=0 then Writeln('Выражение S не определено')
   else begin    {начало блока 1}
          MaxI := A[1];    {первоначальное значение MaxI}
          for I:=2 to 40 do  {цикл 2-для перебора элементов области 1}
            If MaxI<A[I] then MaxI:=A[I]; {проверка условия на максимум}
          MaxJ := A[15];   {первоначальное значение MaxJ}
          for J:=16 to 50 do {цикл 3-для перебора элементов области 2}
            If MaxJ<A[J] then MaxJ:=A[J]; {проверка условия на максимум}
          S := (MaxI + MaxJ)/(MinK + MinL); {вычисление S}
          Writeln('S = ', S:7:2); {вывод на экран значения S}
        end;                          {конец блока 1}
End.                                  {конец программы}
```

Результат работы программы

```
S = -1.00
```

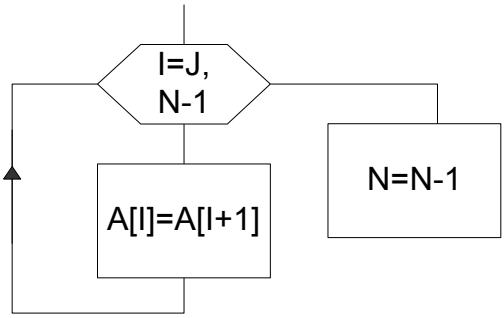
Задача 1.15. Пусть дан массив A, состоящий из 10 элементов, которые вычисляются по закону $a_n = n^2$, где $n = 1; 10$. Разработать схемы алгоритмов и написать программы для следующих подзадач:

а) исключение из массива A седьмого элемента с приведением пошагового словесного алгоритма;

б) исключение из массива A третьего, пятого и восьмого элементов.

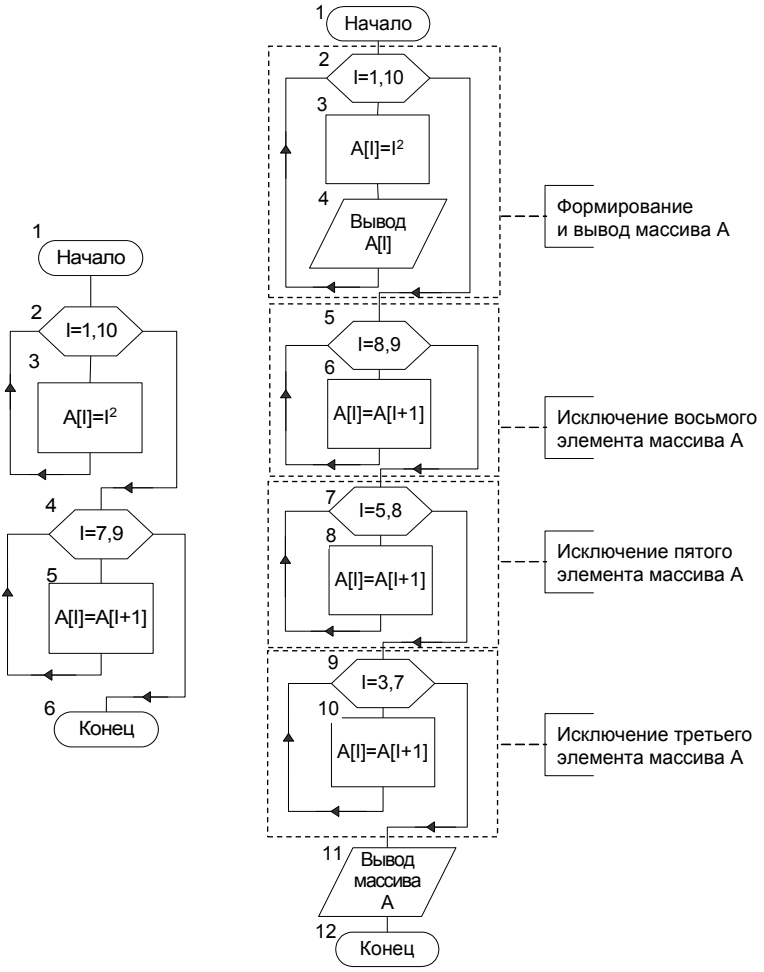
Решение задачи

Алгоритм решения задачи состоит в организации циклов для последовательного переприсваивания порядковых номеров элементам массива A. В общем случае для того, чтобы исключить j-й элемент массива A с числом элементов N, необходимо организовать цикл с параметром $I = J, N–1$.



Для исключения из массива A седьмого элемента необходимо организовать цикл с параметром $I = 7; 9$. Схема алгоритма решения подзадачи 1.15, *а* представлена на рис. 1.15, *а*.

При исключении из массива A нескольких элементов удобнее начинать с элементов, расположенных ближе к концу массива. В нашем случае при решении подзадачи 1.15, *б* будем последовательно исключать восьмой, пятый и третий элементы. Для этого необходимо организовать циклы с параметрами $I = 8; 9$, $I = 5; 8$, $I = 3; 7$. Схема алгоритма решения подзадачи 1.15, *б* представлена на рис. 1.15, *б*.



а *б*

Рис. 1.15

Пояснения к схеме алгоритма (рис. 1.15, а)

Обозначения:

$A[I]$, $A[I + 1]$ – a_i, a_{i+1} элементы массива A.

Символ 1. Начало алгоритма.

Символ 2. Открытие цикла с параметром $I = 1; 10$ для перебора элементов массива A.

Символ 3. Вычисление элемента массива $A[I]$.

Символ 4. Открытие цикла с параметром $I = 7$; 9 для перебора элементов массива A.

Символ 5. Присваивание элементу $A[I]$ значения следующего элемента массива $A[I + 1]$.

Символ 6. Конец алгоритма.

Пояснения к схеме алгоритма (рис. 1.15, б)

Обозначения:

$A[I]$, $A[I + 1] - a_i$, a_{i+1} элементы массива A.

Символ 1. Начало алгоритма.

Символ 2. Открытие цикла с параметром $I = 1$; 10 для перебора элементов массива A.

Символ 3. Вычисление элемента массива $A[I]$.

Символ 4. Вывод на экран элемента массива $A[I]$.

Символ 5. Открытие цикла с параметром $I = 8$; 9 для перебора элементов массива A (после работы цикла в массиве A будет исключен элемент $A[8]$).

Символ 6. Присваивание элементу $A[I]$ значения следующего элемента массива $A[I + 1]$.

Символ 7. Открытие цикла с параметром $I = 5$; 8 для перебора элементов массива A (после работы цикла в массиве будет исключен элемент $A[5]$ исходного массива).

Символ 8. Присваивание элементу $A[I]$ значения следующего элемента массива $A[I + 1]$.

Символ 9. Открытие цикла с параметром $I = 3$; 7 для перебора элементов массива A (после работы цикла в массиве будет исключен элемент $A[3]$ исходного массива).

Символ 10. Присваивание элементу $A[I]$ значения следующего элемента массива $A[I + 1]$.

Символ 11. Вывод на экран массива A производится в цикле (табл. П 1.13), для упрощения вывод представлен в одном символе.

Символ 12. Конец алгоритма.

Пошаговый словесный алгоритм решения подзадачи 1.15, а

Шаг 1. Формирование массива A: 1, 4, 9, 16, 25, 36, 49, 64, 81, 100.

Начало цикла

Шаг 2. Присвоить *I* = 7 (первая итерация цикла).
Шаг 3. *A*[7] = *A*[8] = 64.

Массив A: 1, 4, 9, 16, 25, 36, 64, 64, 81, 100.

Шаг 4. Присвоить *I* = 8 (вторая итерация цикла).
Шаг 5. *A*[8] = *A*[9] = 81.

Массив A: 1, 4, 9, 16, 25, 36, 64, 81, 81, 100.

Шаг 5. Присвоить *I* = 9 (третья итерация цикла).
Шаг 6. *A*[9] = *A*[10] = 100.

Массив A: 1, 4, 9, 16, 25, 36, 64, 81, 100.
Конец цикла

Программа для решения подзадачи 1.15, а

```
Program Prog1_15_1;
Var
   A : Array [1..10] of Real;
   I : Byte;
Begin
   for I := 1 to 10 do  A[I] := I * I; {формирование массива A}
   {элементам массива A от 7 до 9 присвоить значение след. элемента}
   for I := 7 to 9 do  A[I] := A[I+1];
End.
```

Программа для решения подзадачи 1.15, б

```
Program Prog1_15_2;
Var
   A : Array [1..10] of Byte;
   I : Byte;
Begin        {начало раздела операторов программы}
   {формирование и вывод исходного массива A}
   Writeln('Исходный массив');
   for I := 1 to 10 do
   begin
        A[I] := I*I;    Write(A[I]:4, ' ');
   end;
   {исключение восьмого элемента массива A}
```

```
   for I := 8 to 9 do   A[I] := A[I + 1];
   {исключение пятого элемента массива A}
   for I := 5 to 8 do   A[I] := A[I + 1];
   {исключение третьего элемента массива A}
   for I := 3 to 7 do   A[I] := A[I + 1];
   Writeln;  Writeln('Результирующий массив');
   for I := 1 to 7 do  Write(A[I], ' ');  {вывод массива A}
End.          {конец программы}
```

Результат работы программы для решения подзадачи 1.15, б

```
Исходный массив
1    4    9   16   25   36   49   64   81  100
Результирующий массив
1    4   16   36   49   81  100
```

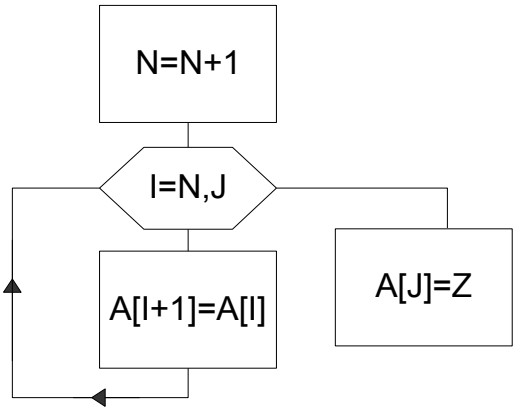
***Задача** 1.16.* Пусть дан массив A, состоящий из 10 элементов, которые вычисляются по закону $a_k = k^2$. Разработать схему алгоритма и программу включения в этот массив дополнительных элементов другого массива, которые вычисляются в зависимости от их порядкового номера по формуле $a_k = \sin(2k)$, для следующих подзадач:

а) включение дополнительного элемента $a_7 = \sin(2 \cdot 7)$ после шестого элемента массива $a_k = k^2$ с приведением пошагового словесного алгоритма;

б) включение дополнительных элементов $a_3 = \sin(2 \cdot 3)$, $a_5 = \sin(2 \cdot 5)$, $a_7 = \sin(2 \cdot 7)$ соответственно после второго, четвертого и шестого элементов исходного массива A.

Решение задачи

Алгоритм состоит из циклов для последовательного переприсваивания порядковых номеров элементам массива A, начиная с последних элементов массива. В общем случае для включения числа Z на j-е место (после $j - 1$ элемента) массива A с числом элементов N необходимо организовать цикл с параметром $I = N, J$.



Для включения в массив *A* седьмого элемента необходимо организовать цикл с параметром *I* = 10; 7. Схема алгоритма решения подзадачи 1.16, *а* представлена на рис. 1.16, *а*. Для включения в массив *A* трёх элементов эту процедуру необходимо повторить трижды. Для этого необходимо организовать циклы с параметрами *I* = 10; 7, *I* = 11; 5, *I* = 12; 3. Схема алгоритма решения подзадачи 1.16, *б* представлена на рис. 1.16, *б*.

Программа для решения подзадачи 1.16, а

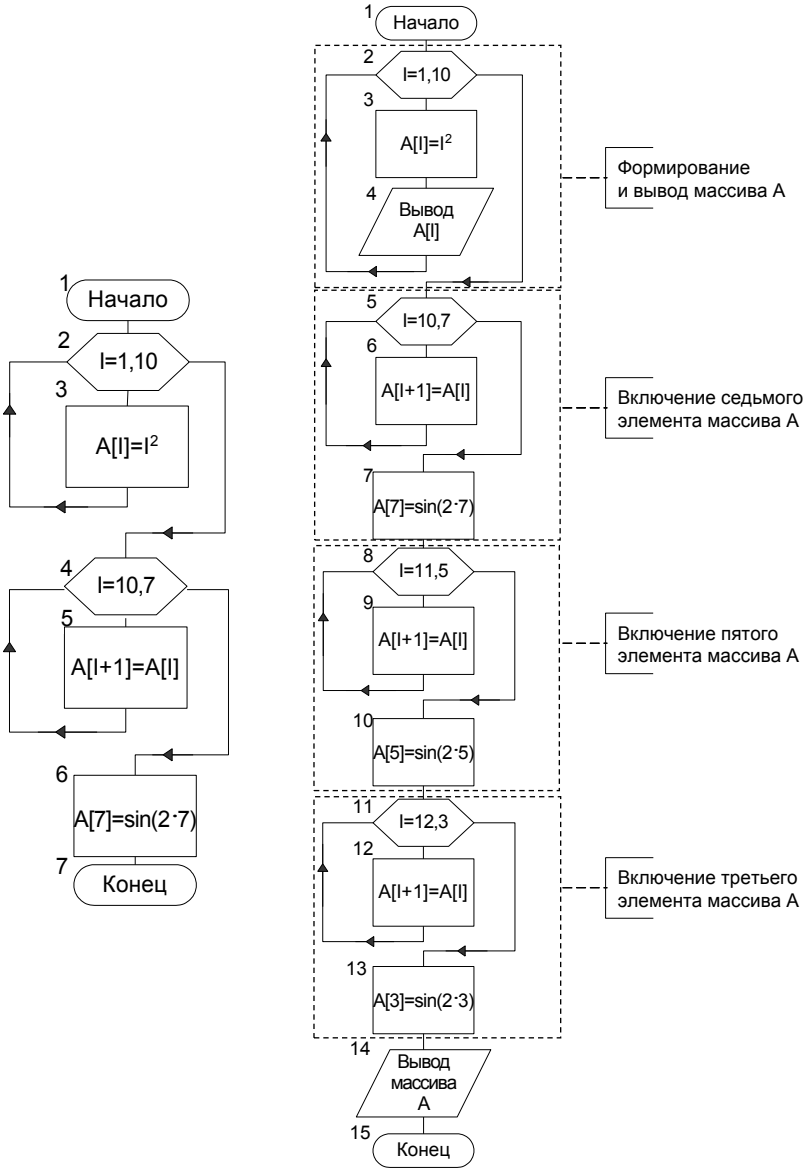
```
Program prog1_16_1;
Var
   A : Array [1..11] of Real;
   I : Byte;
Begin          {начало раздела операторов программы}
   for I := 1 to 10 do  A[I] := I*I; {формирование массива A[I]}
   {элементам массива A от 10 до 7 присвоить знач. предыдущ. элементов}
   for I := 10 downto 7 do  A[I + 1] := A[I];
   A[7] := sin(2*7); {вставка нового значения на 7-е место}
End.                 {конец программы}
```

Пояснения к схеме алгоритма (рис. 1.16, а)

Обозначения

$A[I]$, $A[I + 1] - a_i$, a_{i+1} элементы массива *A*.

Символ 1. Начало алгоритма.


1
Начало
2
I=1,10
3
A[I]=I²
4
Вывод
A[I]
Формирование
и вывод массива A
5
I=10,7
6
A[I+1]=A[I]
7
A[7]=sin(2·7)
Включение седьмого
элемента массива A
8
I=11,5
9
A[I+1]=A[I]
10
A[5]=sin(2·5)
Включение пятого
элемента массива A
11
I=12,3
12
A[I+1]=A[I]
13
A[3]=sin(2·3)
Включение третьего
элемента массива A
14
Вывод
массива
A
15
Конец
1
Начало
2
I=1,10
3
A[I]=I²
4
I=10,7
5
A[I+1]=A[I]
6
A[7]=sin(2·7)
7
Конец

а *б*

Рис. 1.16

Символ 2. Открытие цикла с параметром $I = 1; 10$ для перебора элементов массива A.

Символ 3. Вычисление элемента массива $A[I]$.

Символ 4. Открытие цикла с параметром $I = 10; 7$ для перебора элементов массива A.

Символ 5. Присваивание элементу $A[I + 1]$ значения предыдущего элемента $A[I]$.

Символ 6. Присваивание элементу $A[7]$ значения $\sin(2 \cdot 7)$.

Символ 7. Конец алгоритма.

Пояснения к схеме алгоритма (рис. 1.16, б)

Обозначения:

$A[I]$, $A[I + 1] - a_i$, a_{i+1} элементы массива A.

Символ 1. Начало алгоритма.

Символ 2. Открытие цикла с параметром $I = 1; 10$ для перебора элементов массива A.

Символ 3. Вычисление элемента массива $A[I]$.

Символ 4. Вывод на экран элемента массива $A[I]$.

Включение седьмого элемента

Символ 5. Открытие цикла с параметром $I = 10; 7$ для перебора элементов массива A.

Символ 6. Присваивание элементу $A[I + 1]$ значения предыдущего элемента $A[I]$.

Символ 7. Присваивание элементу $A[7]$ значения $\sin(2 \cdot 7)$.

Включение пятого элемента

Символ 8. Открытие цикла с параметром $I = 11; 5$ для перебора элементов массива A.

Символ 9. Присваивание элементу $A[I + 1]$ значения предыдущего элемента $A[I]$.

Символ 10. Присваивание элементу $A[5]$ значения $\sin(2 \cdot 5)$.

Включение третьего элемента

Символ 11. Открытие цикла с параметром $I = 12; 3$ для перебора элементов массива A.

Символ 12. Присваивание элементу $A[I + 1]$ значения предыдущего элемента $A[I]$.

Символ 13. Присваивание элементу $A[3]$ значения $\sin(2 \cdot 3)$.

Символ 14. Вывод на экран массива A.

Символ 15. Конец алгоритма.

Пошаговый словесный алгоритм решения подзадачи 1.16, а

Шаг 1. Формирование массива A: 1, 4, 9, 16, 25, 36, 49, 64, 81, 100.

```
Начало цикла
```

Шаг 2. Присвоить $I = 10$ (первая итерация цикла).
Шаг 3. $A[11] = A[10] = 100$.

```
Массив A: 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 100.
```

Шаг 4. Присвоить $I = 9$ (вторая итерация цикла).
Шаг 5. $A[10] = A[9] = 81$.

```
Массив A: 1, 4, 9, 16, 25, 36, 49, 64, 81, 81, 100.
```

Шаг 6. Присвоить $I = 8$ (третья итерация цикла).
Шаг 7. $A[9] = A[8] = 64$.

```
Массив A: 1, 4, 9, 16, 25, 36, 49, 64, 64, 81, 100.
```

Шаг 8. Присвоить $I = 7$ (четвертая итерация цикла).
Шаг 9. $A[8] = A[7] = 49$.

```
Массив A: 1, 4, 9, 16, 25, 36, 49, 49, 64, 81, 100.
Конец цикла
```

Шаг 10. Так как $I = 7$, то $A[7] = \sin(2 \cdot 7) = \sin(14)$.

```
Массив A: 1, 4, 9, 16, 25, 36, sin(14), 49, 64, 81, 100.
```

Программа для решения подзадачи 1.16, б

```
Program prog1_16_2;
Var
   A : Array [1..13] of Real;
   I : Byte;
Begin         {начало раздела операторов программы}
   {формирование и вывод исходного массива A}
   Writeln('Исходный массив');
   for I := 1 to 10 do
   begin
     A[I] := I*I;    Write(A[I]:4:0, ' ');
   end;
   {включение дополнительного седьмого элемента}
   for I := 10 downto 7 do   A[I + 1] := A[I];
   A[7] := sin(2*7);  {вставка нового значения на 7-е место}
   {включение дополнительного пятого элемента}
   for I := 11 downto 5 do   A[I + 1] := A[I];
   A[5] := sin(2*5);  {вставка нового значения на 5-е место}
   {включение дополнительного третьего элемента}
   for I := 12 downto 3 do   A[I + 1] := A[I];
   A[3] := sin(2*3);  {вставка нового значения на 3-е место}
   Writeln;  Writeln('Результирующий массив');
   for I := 1 to 13 do  Write(A[I]:5:1, ' '); {вывод массива A}
End.   {конец программы}
```

Результат работы программы для решения подзадачи 1.16, б

```
Исходный массив
1    4    9   16   25   36   49   64   81  100
Результирующий массив
1.0  4.0 -0.3  9.0 16.0 -0.5 25.0 36.0  1.0 49.0 64.0 81.0 100.0
```

Задача 1.17. Разработать схему алгоритма и программу получения булевой матрицы B из матрицы A размером 10×10, элементы которой формируются по закону $a_{ij} = \sin(i + j)$. При формировании булевой матрицы использовать пороговое значение $k = 0{,}3$. Значения исходной и полученной матриц вывести на экран.

Решение задачи

Элементы булевой матрицы B формируются по закону

$$b_{ij} = \begin{cases} 0, \text{ если } |a_{ij}| \le k; \\ 1, \text{ если } |a_{ij}| > k. \end{cases}$$

Схема алгоритма решения задачи представлена на рис. 1.17.

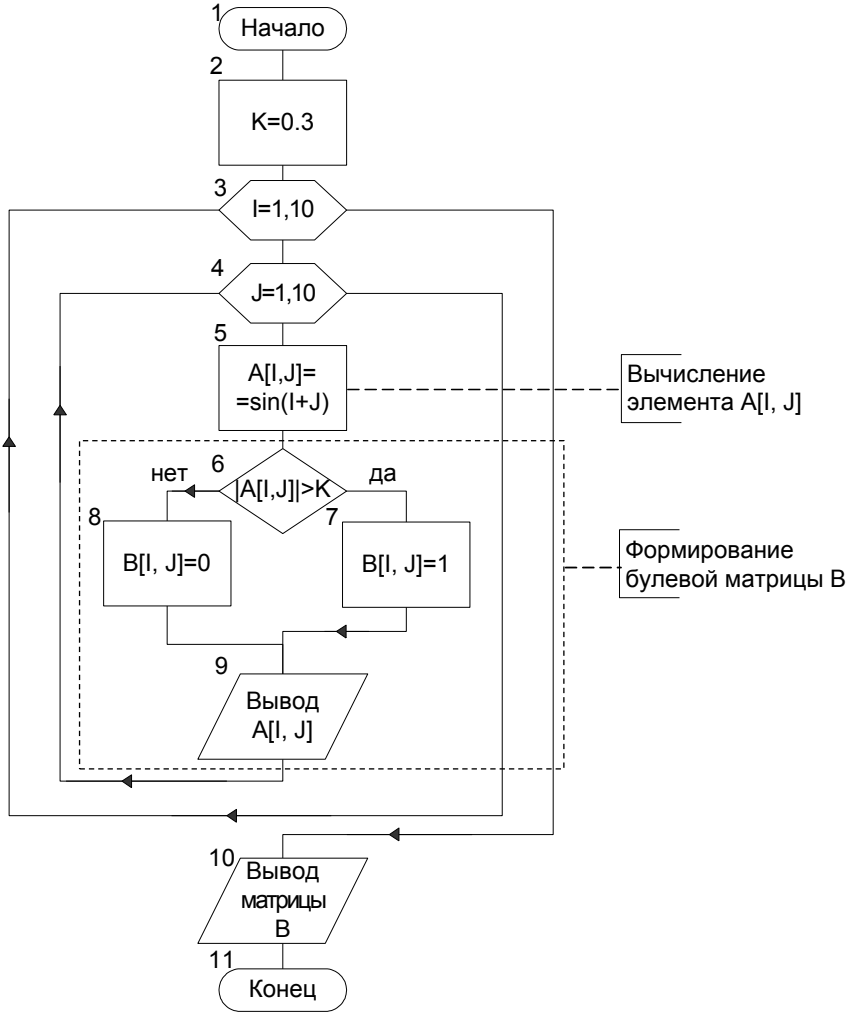


Рис. 1.17

Пояснения к схеме алгоритма

Обозначения:

$A[I, J] – a_{ij}$ элемент матрицы A;

$B[I, J] – b_{ij}$ элемент матрицы B.

Символ 1. Начало алгоритма.

Символ 2. Присвоение переменной K значения 0,3.

Символ 3. Открытие цикла с параметром $I = 1; 10$ для перебора строк матрицы A.

Символ 4. Открытие цикла с параметром $J = 1; 10$ для перебора столбцов матрицы A.

Символ 5. Вычисление элемента матрицы $A[I, J]$.

Символ 6. Проверка значения $A[I, J]$. Если значение больше порогового значения K, то выполняется *символ 7*, если нет – *символ 8*.

Символ 7. Присвоение элементу матрицы $B[I, J]$ значения 1.

Символ 8. Присвоение элементу матрицы $B[I, J]$ значения 0.

Символ 9. Вывод на экран элемента $A[I, J]$.

Символ 10. Вывод матрицы B. Вывод значений производится с организацией внешнего цикла для перебора строк и внутреннего цикла для перебора столбцов матрицы (табл. П 1.13), на схеме для упрощения это показано в одном символе.

Символ 11. Конец алгоритма.

Программа

```
Program Prog1_17;
Var
   K : Real;
   A, B : Array[1..10, 1..10] of Real;
   I, J : Byte;
Begin           {начало раздела операторов программы}
   K := 0.3 ;   {пороговое значение}
   {формирование матриц A и B }
   for I := 1 to 10 do
   begin     {начало цикла 1 - для перебора строк матрицы A}
     for J := 1 to 10 do
     begin   {начало цикла 2 - для перебора столбцов матрицы A}
       A[I, J] := sin(I + J);  вычисление элемента A[I, J]}
       If Abs(A[I, J]) > K then B[I, J] := 1 else B[I, J] := 0;
       Write(A[I, J]:4:1, ' ');    {вывод элемента A[I, J]}
     end;                          {конец цикла 2}
```

```
    Writeln;                {переход на след. строку}
  end;                      {конец цикла 1}
  {вывод матрицы B}
  for I := 1 to 10 do
  begin      {начало цикла 3 - для перебора строк матрицы A}
    for J := 1 to 10 do Write(B[I, J]:4:0, ' ');
    Writeln;                {переход на след. строку}
  end;                      {конец цикла 3}
End.                        {конец программы}
```

Результат работы программы

Исходная матрица

```
 0.9  0.1 -0.8 -1.0 -0.3  0.7  1.0  0.4 -0.5 -1.0
 0.1 -0.8 -1.0 -0.3  0.7  1.0  0.4 -0.5 -1.0 -0.5
-0.8 -1.0 -0.3  0.7  1.0  0.4 -0.5 -1.0 -0.5  0.4
-1.0 -0.3  0.7  1.0  0.4 -0.5 -1.0 -0.5  0.4  1.0
-0.3  0.7  1.0  0.4 -0.5 -1.0 -0.5  0.4  1.0  0.7
 0.7  1.0  0.4 -0.5 -1.0 -0.5  0.4  1.0  0.7 -0.3
 1.0  0.4 -0.5 -1.0 -0.5  0.4  1.0  0.7 -0.3 -1.0
 0.4 -0.5 -1.0 -0.5  0.4  1.0  0.7 -0.3 -1.0 -0.8
-0.5 -1.0 -0.5  0.4  1.0  0.7 -0.3 -1.0 -0.8  0.1
-1.0 -0.5  0.4  1.0  0.7 -0.3 -1.0 -0.8  0.1  0.9
```

Полученная булева матрица

```
   1    0    1    1    0    1    1    1    1    1
   0    1    1    0    1    1    1    1    1    1
   1    1    0    1    1    1    1    1    1    1
   1    0    1    1    1    1    1    1    1    1
   0    1    1    1    1    1    1    1    1    1
   1    1    1    1    1    1    1    1    1    0
   1    1    1    1    1    1    1    1    0    1
   1    1    1    1    1    1    1    0    1    1
   1    1    1    1    1    1    0    1    1    0
   1    1    1    1    1    0    1    1    0    1
```

2. ЗАДАЧИ ПОВЫШЕННОЙ СЛОЖНОСТИ ПО СОСТАВЛЕНИЮ АЛГОРИТМОВ И ПРОГРАММ

2.1. Задачи с элементами теории множеств

Задача 2.1. Разработать схему алгоритма и программу для определения количества точек (x, y), удовлетворяющих системе неравенств

$$\begin{cases} 3x + 4y < 41{,}5; \\ 5x - y > 1{,}2; \\ xy > x + y. \end{cases}$$

Все используемые точки (x, y) должны удовлетворять множеству, определяемому по правилам x = 1; 20 (0,1), y = 0; 10 (0,2). Число комбинаций точек вывести на экран.

Решение задачи

При решении задачи следует использовать табуляцию функции (п. 1.4). Схема алгоритма решения задачи представлена на рис. 2.1.

Обозначение схемы алгоритма

Count – число точек, удовлетворяющих системе неравенств.

Программа

```
Program Task2_1;
Var
   X, Y  : Real;
   Count : Integer;
Begin                {начало раздела операторов программы}
   Count := 0; X:=1; {начальные значения переменных Count, X}
   While X <= 20 do
   Begin      {начало цикла 1 - для перебора значений X}
     Y:=0;    {начальное значение переменной Y}
     While Y <= 10 do
     Begin   {начало цикла 2 - для перебора значений Y}
       {если X, Y удовлетворяют условию, Count увеличивается на 1}
       If (3*X+4*Y < 41.5) and (5*X-Y > 1.2) and (X*Y > X+Y)
```

```
                        Then Count:=Count + 1;
        Y:= Y + 0.2;          {приращение переменной Y}
      end;                    {конец цикла 2}
      X:= X + 0.1;            {приращение переменной X}
    end;                      {конец цикла 1}
    {вывод результата на экран}
    Writeln('Число комбинаций точек (x, y) равно ', Count);
    Readln;                   {ожидание нажатия клавиши Enter}
End.                          {конец программы}
```

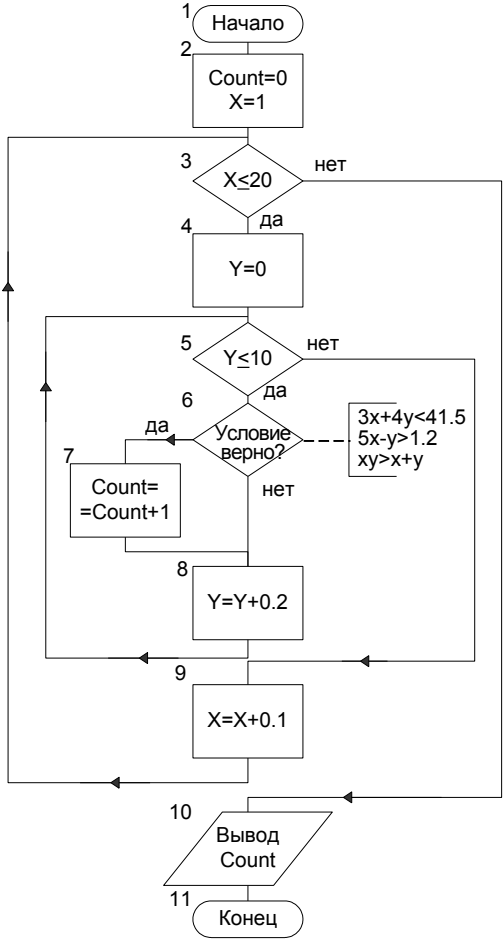


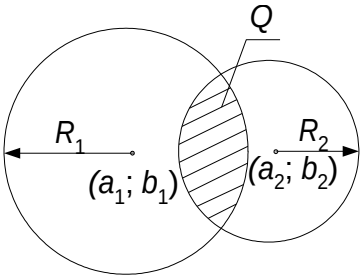
Рис. 2.1

Результат работы программы

```
Число комбинаций точек (x, y) равно 2108
```

Задача 2.2. Разработать схему алгоритма и программу вычисления и вывода на экран общего количества точек $(x, y) \in Q$, принадлежащих одновременно двум окружностям, заданным системой неравенств

$$Q = \begin{cases} (x-a_1)^2 + (y-b_1)^2 \le R_1^2; \\ (x-a_2)^2 + (y-b_2)^2 \le R_2^2. \end{cases}$$



Аргументы x и y изменяются по законам x = 0; 200 (0,1), y = 0; 300 (0,2). С клавиатуры вводятся координаты центров кругов (a_1, b_1) и (a_2, b_2) и значения радиусов R_1, R_2.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.2.

Обозначение схемы алгоритма

Count – количество точек (x, y), принадлежащих первому и второму кругу.

Программа

```
Program Task2_2;
Uses Crt;
```

```
Var
   a1, a2, b1, b2, R1, R2, x, y : Real;
   Count : Word;
Begin                {начало раздела операторов программы}
   ClrScr;           {очистка экрана}
   {ввод данных}
   Write('Введите координаты центров кругов a1,a2,b1,b2: ');
   Readln(a1, a2, b1, b2);
   Write('Введите радиусы кругов R1, R2: '); Readln(R1, R2);
   {расчет}
   x:= 0; Count:= 0; {первоначальные значения переменных x, Count}
   While x <= 200 do
   Begin             {начало цикла 1 – для перебора значений x}
     y:= 0;          {первоначальное значение y}
     While y <= 300 do
     Begin    {начало цикла 2 – для перебора значений y}
       {проверка условия принадлежности точки (x, y) кругам}
       If ((x-a1)*(x-a1) + (y-b1)*(y-b1) <= R1*R1) and
          ((x-a2)*(x-a2) + (y-b2)*(y-b2) <= R2*R2)
                                          Then Count:=Count+1;
       y:= y + 0.2; {приращение переменой y}
     end;           {конец цикла 2}
     x:= x + 0.1;   {приращение переменой x}
   end;             {конец цикла 1}
   {вывод результата на экран}
   Write('Число точек, принадлежащих двум кругам = ', Count);
   Readln;          {ожидание нажатия клавиши Enter}
End.                {конец программы}
```

Результат работы программы

```
Введите координаты центров кругов a1, a2, b1, b2: 1 2 3 4
Введите радиусы кругов R1, R2: 10 20
Число точек, принадлежащих двум кругам = 6130
```

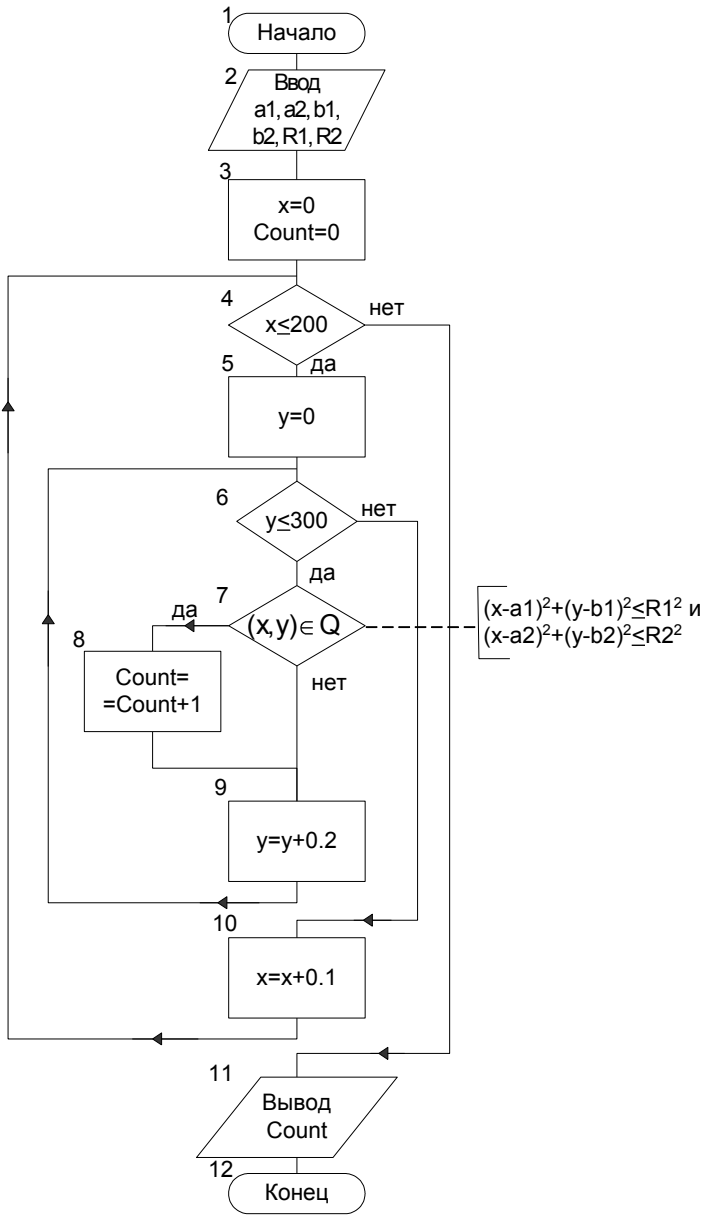

1
Начало
2
Ввод
a1, a2, b1,
b2, R1, R2
3
x=0
Count=0
4
x≤200
нет
5
да
y=0
6
y≤300
нет
да
7
да
(x,y)∈ Q
(x-a1)²+(y-b1)²≤R1² и
(x-a2)²+(y-b2)²≤R2²
8
Count=
=Count+1
нет
9
y=y+0.2
10
x=x+0.1
11
Вывод
Count
12
Конец

Рис. 2.2

2.2. **Задачи табуляции функций нескольких переменных**

Задача 2.3. Разработать схемы алгоритмов и программы табуляции функции $z = x^2 + y^3$, в которой переменные x, y изменяются в различных математических формах.

а) Переменные x, y изменяются следующим образом:

$$x = x_n; x_k (h_1),$$

$$y = y_n; y_k (h_2),$$

где x_n, y_n – начальные значения переменных x, y;

x_k, y_k – конечные значения переменных x, y;

h_1, h_2 – шаги изменения переменных x, y.

Значения x_n, y_n, x_k, y_k, h_1, h_2 ввести с клавиатуры. Значения x, y, z вывести на экран.

б) Переменные x, y заданы в виде массивов размерами n_1 и n_2 элементов. Размеры и значения массивов ввести с клавиатуры. Значения функции z представить в виде массива. На экран вывести значения x, y, z.

в) Переменные x, y заданы смешанными законами, а именно, x – одномерный массив размером n, а переменная y изменяется с заданным шагом $y = y_n; y_k (h)$, где y_n, y_k – начальное и конечное значения переменной y; h – шаг изменения переменной y. Значения y_n, y_k, h, а также значения элементов и размер массива x ввести с клавиатуры. При написании программы использовать оператор *for*. Значения x, y, z вывести на экран в виде таблицы.

Решение задачи

Схемы алгоритмов решений подзадач 2.3, *а*, 2.3, *б*, 2.3, *в* представлены соответственно на рис. 2.3, *а–в*.

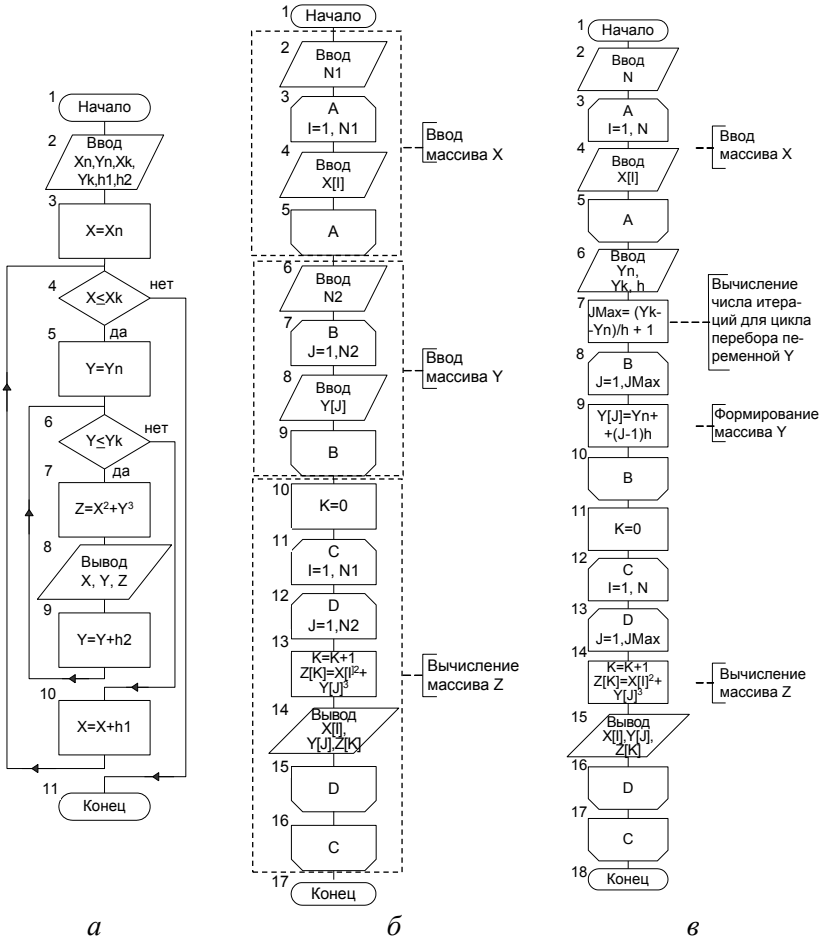


Рис. 2.3

Обозначения схем алгоритмов

K – дополнительная переменная для индексирования результирующего массива Z;

$X[I]$ – x_i элемент массива X;

$Y[J]$ – y_j элемент массива Y;

$Z[K]$ – z_k элемент массива Z;

JMax – число итераций для цикла, в котором производится перебор значений переменной *Y*.

Программа для решения подзадачи 2.3, а

```
Program Task2_3a;
Var
   Xn, Yn, Xk, Yk, h1, h2 : Real;
   X, Y, Z                : Real;
Begin
   {ввод значений для перебора переменных X и Y}
   Write('Введите начальные значения переменных X и Y: ');
   Readln(Xn, Yn);
   Write('Введите конечные значения переменных X и Y: ');
   Readln(Xk, Yk);
   Write('Введите шаги изменения переменных X и Y: ');
   Readln(h1, h2);
   {расчет}
   X:= Xn;           {начальное значение переменной X}
   While X <= Xk do
   Begin      {начало цикла 1 - для перебора значений X}
     Y:= Yn;         {начальное значение переменной Y}
     While Y <= Yk do
     Begin    {начало цикла 2 - для перебора значений Y}
       Z:= X*X + Y*Y*Y;    {вычисление Z}
       {вывод результатов}
       Writeln('X = ', X:4:2, ' Y = ', Y:4:2, ' Z = ', Z:7:3);
       Y:= Y + h2;  {приращение переменной Y}
     end;           {конец цикла 2}
     X:= X + h1;    {приращение переменной X}
   end;             {конец цикла 1}
   Readln;          {ожидание нажатия клавиши Enter}
End.                {конец программы}
```

Результат работы программы

```
Введите начальные значения переменных X и Y: 1 3
Введите конечные значения переменных X и Y: 5 4
Введите шаги изменения переменных X и Y : 2 0.5
X = 1.00 Y = 3.00 Z =  28.000
X = 1.00 Y = 3.50 Z =  43.875
X = 1.00 Y = 4.00 Z =  65.000
X = 3.00 Y = 3.00 Z =  36.000
X = 3.00 Y = 3.50 Z =  51.875
X = 3.00 Y = 4.00 Z =  73.000
```

```
X = 5.00 Y = 3.00 Z =  52.000
X = 5.00 Y = 3.50 Z =  67.875
X = 5.00 Y = 4.00 Z =  89.000
```

Программа для решения подзадачи 2.3, б

```
Program Task2_3b;
Var
   X, Y : Array [1..20] of Real;
   Z    : Array [1..400] of Real;
   N1, N2, I, J, K :Integer;
Begin         {начало раздела операторов программы}
   {ввод значений}
   Write('Введите размер массива X (<=20): ');  Readln(N1);
   Write('Введите значения массива X: ');
   for I := 1 to N1 do Read(X[I]); {ввод массива X}
   Write('Введите размер массива Y (<=20): ');  Readln(N2);
   Write('Введите значения массива Y: ');
   for J := 1 to N2 do   Read(Y[J]); {ввод массива Y}
   {расчет}
   K:=0;       {первоначальное значение индекса массива Z}
   for I:=1 to N1 do  {цикл 1 - для перебора значений X}
     for J:=1 to N2 do{цикл 2 - для перебора значений Y}
     begin           {начало циклов 1, 2}
       K:= K + 1;    {увеличение индекса массива Z на 1}
       Z[K]:=X[I]*X[I] + Y[J]*Y[J]*Y[J]; {вычисление функции Z}
       {вывод результатов на экран}
       Writeln('X = ',X[I]:4:2,' Y = ',Y[J]:4:2,' Z = ',Z[K]:6:2);
     end;            {конец цикла 2, конец цикла 1}
End.                 {конец программы}
```

Результат работы программы

```
Введите размер массива X (число <=20): 3
Введите значения массива X: 1 3 5
Введите размер массива Y (число <=20): 3
Введите значения массива Y: 3 3.5 4
X = 1.00 Y = 3.00 Z =  28.00
X = 1.00 Y = 3.50 Z =  43.87
X = 1.00 Y = 4.00 Z =  65.00
X = 3.00 Y = 3.00 Z =  36.00
X = 3.00 Y = 3.50 Z =  51.88
X = 3.00 Y = 4.00 Z =  73.00
X = 5.00 Y = 3.00 Z =  52.00
X = 5.00 Y = 3.50 Z =  67.87
X = 5.00 Y = 4.00 Z =  89.00
```

Программа для решения подзадачи 2.3, в

```
Program Task2_3c;
Var
  Yn, Yk, h, Y, Z  : Real;
  N, I, J, JMax, K  : Integer;
  X  : Array [1..100] of Real;
Begin                {начало раздела операторов программы}
  {ввод значений}
  Write('Введите размер массива X (<=100): ');  Readln(N);
  Write('Введите значения элементов массива X: ');
  for I:=1 to N do Read(X[I]);{ввод значений элементов массива X}
  Write('Введите начальное, конечное значения и шаг изменения Y: ');
  Readln(Yn, Yk, h);
  {расчет}
  JMax:= Trunc((Yk - Yn)/h + 1); {число итераций для цикла 2}
  K:=0;
  {вывод названий столбцов на экран}
  Writeln('|-----------------------------|');
  Writeln('|Номер|   X   |   Y   |   Z    |');
  Writeln('|-----------------------------|');
  for I:=1 to N do {начало цикла 1-для перебора значений X}
     for J:=1 to JMax do
     Begin          {начало цикла 2- для перебора значений Y}
       K:= K + 1;
       Y:= Yn + (J-1)*h;          {вычисление переменной Y}
       Z:= X[I]*X[I] + Y*Y*Y;     {вычисление функции Z}
       {вывод результатов на экран}
       Writeln('|', K:4,' |',X[I]:6:2,' |',Y:6:2,' |',Z:7:2,' |');
     end;           {конец цикла 1, конец цикла 2}
  Writeln('|-----------------------------|');
  Readln;           {ожидание нажатия клавиши Enter}
End.                {конец программы}
```

Результат работы программы

```
Введите размер массива X (<=100): 3
Введите значения элементов массива X: 1 3 5
Введите начальное, конечное значения и шаг изменения Y: 3 4 0.5
|-----------------------------|
|Номер|   X   |   Y   |   Z    |
|-----------------------------|
|   1 |  1.00 |  3.00 |  28.00 |
|   2 |  1.00 |  3.50 |  43.87 |
|   3 |  1.00 |  4.00 |  65.00 |
|   4 |  3.00 |  3.00 |  36.00 |
```

```
|    5 |  3.00 |  3.50 |  51.88 |
|    6 |  3.00 |  4.00 |  73.00 |
|    7 |  5.00 |  3.00 |  52.00 |
|    8 |  5.00 |  3.50 |  67.87 |
|    9 |  5.00 |  4.00 |  89.00 |
|-------------------------------|
```

Задача 2.4. Разработать схему алгоритма и программу табуляции функции $z = \log_3(x+y) + \sqrt{x-y}$ с ограниченной областью определения, где $x = 1; 200$ (1) и $y = 40; 100$ (2). Если функцию z вычислить нельзя в силу области ее определения, то принять $z = \cos(x + y)$. Вывести в файл *a.a* все значения z, для которых выполняется условие

$$\begin{cases} z > 10; \\ x > 31\sqrt{y}. \end{cases}$$

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.4.

Программа

```
Program Task2_4;
Var
  Z    : Real;
  X, Y : Byte;
  f    : Text;
Begin        {начало раздела операторов программы}
  Assign(f, 'a.a');  Rewrite(f);  {создание файла a.a}
  for X := 1 to 200 do
  Begin      {начало цикла 1 - для перебора значений X}
    Y := 40; {начальное значение Y}
    While Y <= 100 do
    Begin    {начало цикла 2 - для перебора значений Y}
      If (X - Y) >= 0 Then Z := Ln(X+Y)/Ln(3) + Sqrt(X-Y)
                      Else Z := Cos(X+Y);
      {если условие выполняется, то вывод в файл}
      If (Z > 10) and (X > 31*Sqrt(Y)) Then
               Writeln(f, 'X = ',X,' Y = ',Y,' Z = ',Z:6:2);
      Y := Y + 2;  {приращение переменной Y}
    End;           {конец цикла 2}
```

```
  End;                          {конец цикла 1}
  Close(f);                     {закрытие файла a.a}
End.                            {конец программы}
```

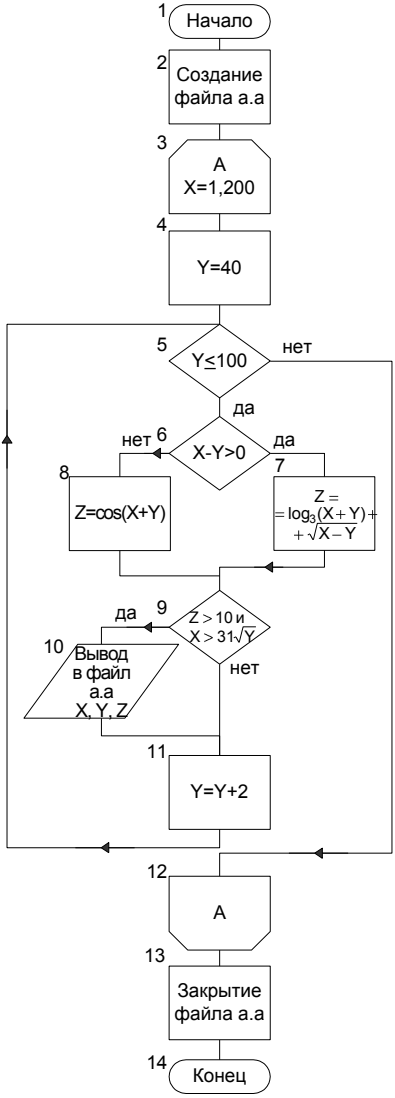


Рис. 2.4

Обозначение схемы алгоритма

$X[I] - x_i$ элемент массива X.

Результат работы программы

Файл a.a содержит:

```
X = 197 Y = 40 Z =  17.51
X = 198 Y = 40 Z =  17.55
X = 199 Y = 40 Z =  17.59
X = 200 Y = 40 Z =  17.64
```

***Задача** 2.5.* Разработать схему алгоритма и программу вычисления значения функции $y = \lg(\sin 5x)^3$ для x = 40,1; 50,2; 70,8, находящихся в файле *a.a.* Вывести на экран значения x и y. Если функция y не имеет значения в силу области определения x, вывести сообщение «Значение y не определено».

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.5.

Программа

```
Program Task2_5;
Var
   I    : Integer;
   F    : Text;
   X    : Array [1..3] of Real;
   Y    : Real;
Begin        {начало раздела операторов программы}
   Assign(F, 'a.a'); Reset(F); {открытие файла a.a}
   for I:=1 to 3 do
   Begin                    {начало цикла}
     Read(F, X[I]);         {чтение значения X[I] из файла}
     If Sin(5*X[I])>0 Then {если X принадлежит области определения}
     Begin
       Y := Sin(5*X[I])*Sin(5*X[I])*Sin(5*X[I]);
       Y:=Ln(Y)/Ln(10);     {вычисление Y}
       Writeln('При X = ',X[I]:3:1,' Y = ',Y:5:2);{вывод Y на экран}
     End Else Writeln('При X = ',X[I]:3:1,' значение Y не определено');
```

```
   End;       {конец цикла 1}
   Close(F); {закрытие файла a.a}
   Readln;   {ожидание нажатия клавиши Enter}
End.          {конец программы}
```

Результат работы программы

```
При X = 40.1 значение Y не определено
При X = 50.2 значение Y не определено
При X = 70.8 Y = -0.22
```

2.3. Задачи с использованием массивов различной размерности

Задача 2.6. Разработать схему алгоритма и программу вычисления выражения $S = \max\{a_i\} + \min\{a_j\}$, где $i = 2, 4, 6, \ldots, 100$ – четные индексы элементов массива A, $j = 1, 3, 5, \ldots, 99$ – нечетные индексы элементов массива A. Элементы массива A формируется по закону $a_k = \cos(k^2 + 2k + 30)$. Результат вычисления вывести на экран.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.6.

Обозначения схемы алгоритма

$A[K]$ – a_k элемент массива A;

Max – значение максимального элемента среди четных элементов массива A;

Min – значение минимального элемента среди нечетных элементов массива A.

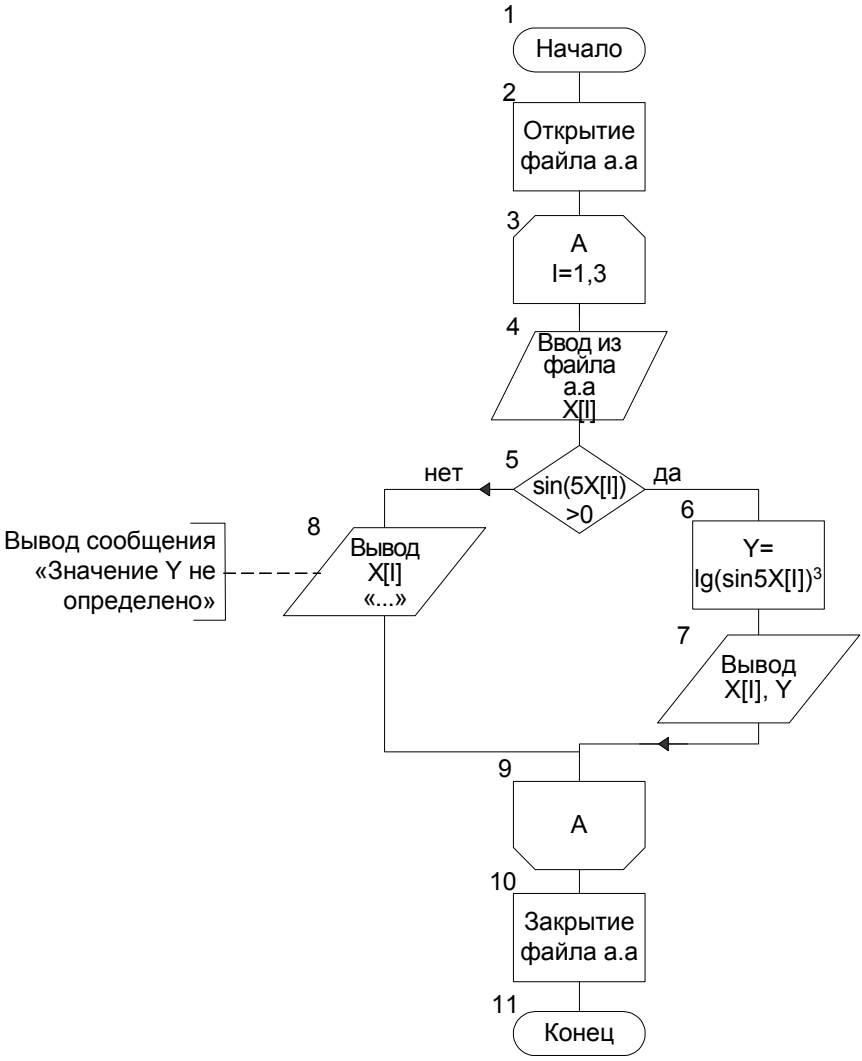

1
Начало
2
Открытие файла a.a
3
A
I=1,3
4
Ввод из файла a.a X[I]
5
sin(5X[I]) >0
нет
да
8
Вывод X[I] «...»
Вывод сообщения «Значение Y не определено»
6
Y= lg(sin5X[I])3
7
Вывод X[I], Y
9
A
10
Закрытие файла a.a
11
Конец

Рис. 2.5

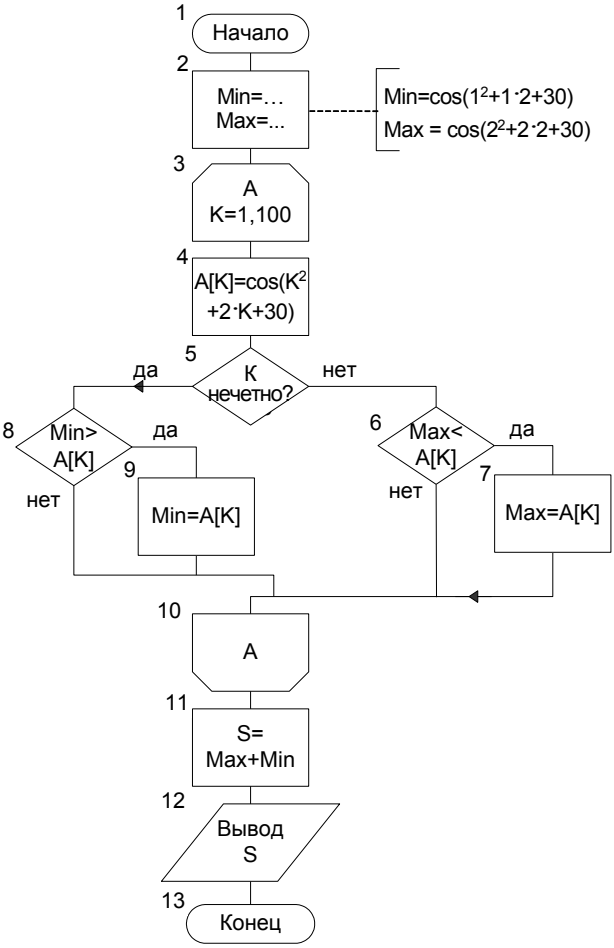


Рис. 2.6

Программа

```
Program Task2_6;
Uses Crt;
Var
   Min, Max, S : Real;
   K : Integer;
   A : Array[1..100] of Real;
```

```
Begin                    {начало раздела операторов программы}
   ClrScr;               {очистка экрана}
   Min:=cos(1+2+30);     {начальное значение переменной минимума}
   Max:=cos(2*2+2*2+30);{начальное значение переменной максимума}
   for K := 1 to 100 do
   begin                 {начало цикла}
     A[K]:=cos(K*K + 2*K + 30); {вычисление элемента A[K]}
     If odd(K) Then              {проверка на нечетность}
        If Min>A[K] then Min:=A[K] {если нечетный элемент}
                    Else If Max<A[K] then Max:=A[K]; {если четный элемент}
   end;                  {конец цикла}
   S:=Max + Min; {сумма максимального и минимального элементов}
   Writeln('Сумма максимального и минимального элементов
массива = ', S:8:5); {вывод S на экран}
   Readln;               {ожидание нажатия клавиши Enter}
End.                     {конец программы}
```

Результат работы программы

```
Сумма максимального и минимального элементов массива = -0.00303
```

***Задача** 2.7.* Разработать схему алгоритма и программу для нахождения максимального элемента каждой строки матрицы A. Элементы матрицы формируются по закону $a_{ij} = \cos(i + j)$, $i = 1; 10$, $j = 1; 30$. Максимальный элемент вычислять с использованием подпрограммы (*function*). Максимальные значения выводить с точностью до пяти знаков после десятичной точки.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.7.

Обозначения схемы алгоритма

$A[I, J]$ – a_{ij} элемент матрицы A;

$B[J]$ – b_j элемент одномерного массива B, который формируется из строки матрицы A;

MaxB(B) – вызов функции *MaxB* вычисления максимального элемента одномерного массива B;

Max – значение максимального элемента массива B, равное возвращаемому значению функции *MaxB*.

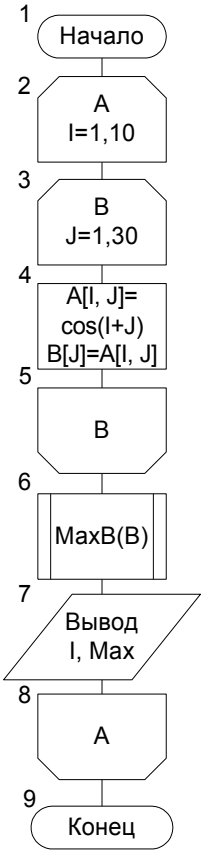


Рис. 2.7

Программа

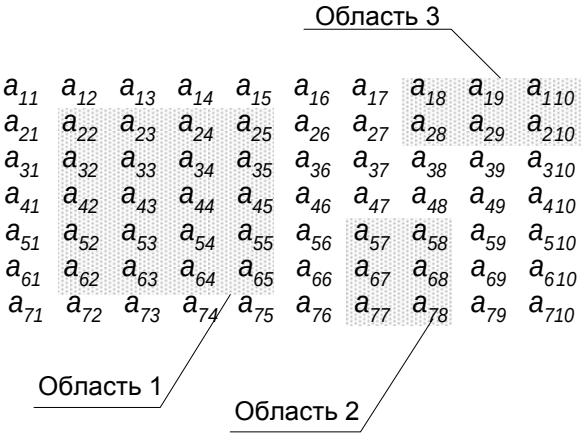
```
Program Task2_7;
Type
   TypeB = array[1..30] of Real; {описание типа массива TypeB}
Var
   A : Array[1..10, 1..30] of Real;
   B : TypeB;
   I, J : Integer;
   Max : Real;
{функция нахождения максимального элемента массива B1}
Function MaxB(B1 : TypeB) : Real;
```

```
var
   I : Integer;
   Max : Real;
begin              {начало раздела операторов функции MaxB}
   Max := B1[1];  {начальное значение переменной максимума}
   for I := 2 to 30 do
      If Max < B1[I] then Max := B1[I];
   MaxB := Max;
end;           {конец функции MaxB }
Begin          {начало раздела операторов основной программы}
   for I := 1 to 10 do
   begin       {начало цикла 1-для перебора строк массива A}
      for J := 1 to 30 do
      begin {начало цикла 2-для перебора столбцов массива B}
         A[I, J] := cos(I * J);{вычисление элемента массива A}
         B[J] := A[I, J]; {создание массива B из строки массива A}
      end;                    {конец цикла 2}
      Max := MaxB(B);         {вызов функции MaxB}
      {вывод на экран}
      Writeln('Максимальный элемент ',I,'-й строки = ',Max:8:5);
   end;                       {конец цикла 1}
end.                          {конец основной программы}
```

Результат работы программы

```
Максимальный элемент 1-й строки =  0.99120
Максимальный элемент 2-й строки =  0.99984
Максимальный элемент 3-й строки =  0.99339
Максимальный элемент 4-й строки =  0.99984
Максимальный элемент 5-й строки =  0.99120
Максимальный элемент 6-й строки =  0.99859
Максимальный элемент 7-й строки =  0.98590
Максимальный элемент 8-й строки =  0.99937
Максимальный элемент 9-й строки =  0.98590
Максимальный элемент 10-й строки =  0.99609
```

***Задача** 2.8.* Разработать схему алгоритма и программу нахождения минимального элемента двухмерного массива *A* размером 7×10, который сформирован из случайных чисел, находящихся в интервале от 0 до 999 включительно. Значение минимума искать в заштрихованных областях двумерного массива. Поиск минимального элемента оформить в виде подпрограммы (*procedure*). Значения элементов массива *A* в заштрихованной области и минимального элемента вывести на экран.



Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.8.

Программа

```
Program Task2_8;
Type
   T1 = array[1..40] of Integer;
Var
   A : Array[1..7, 1..10] of Integer;
   I, J, K, Min : Integer;
   B : T1;
{процедура поиска минимального элемента (MinX) массива X раз-
мерности N}
Procedure MinArr(X : T1; N : Integer; Var MinX : Integer);
Var  I : Integer;
begin         {начало раздела операторов процедуры MinArr}
   MinX := X[1];     {начальное значение переменной минимума}
   for I:=2 to N do {цикл для перебора элементов массива X}
     If MinX > X[I] then MinX := X[I];
end;   {конец процедуры MinArr }
Begin         {начало раздела операторов основной программы}
  K := 0;     {начальное значение числа элементов массива B}
  Writeln('Значения элементов массива заштрихованной области:');
  for I := 1 to 7 do   {цикл 1-для перебора строк массива A}
    for J := 1 to 10 do  {цикл 2-для перебора столбцов массива A}
    begin              {начало циклов 1, 2}
```

```
        A[I, J] := Random(1000);  {вычисление A[I, J]}
        {проверка условия 1}
        If ((I>=2) and (I<=6) and (J>=2) and (J<=5)) or
              {проверка условия 2}
              ((I>=5) and (I<=7) and (J>=7) and (J<=8)) or
              {проверка условия 3}
              ((I>=1) and (I<=2) and (J>=8) and (J<=10)) then
        begin {если A[I,J] находится в заштрихов. области}
          K := K + 1;{число элементов массива B увел. на 1}
          B[K] := A[I, J];
          Write(B[K], ' '); {вывод элемента B[K] на экран}
        end;
      end;                    {конец циклов 1,2}
      MinArr(B, K, Min);      {вызов процедуры MinArr}
      Writeln; Writeln('Минимальный элемент = ',Min); {вывод на экран}
      Readln;                 {ожидание нажатия клавиши Enter}
End.                          {конец основной программы}
```

Обозначения схемы алгоритма

$A[I, J]$ – a_{ij} элемент матрицы A;

$B[K]$ – b_k элемент одномерного массива B, который формируется из элементов заштрихованных областей матрицы A;

K – счетчик элементов массива B;

MinArr(B, K, Min) – процедура определения минимального элемента одномерного массива B, состоящего из K элементов, результат процедуры заносится в переменную *Min*.

Результат работы программы

```
Значения элементов массива заштрихованной области:
161 371 425 474 70 840 59 367 773 327 843 717 306 162 149
873 287 771 143 500 21 592 773 650 205 680 592 954 997 243
952 684.
Минимальный элемент = 21
```

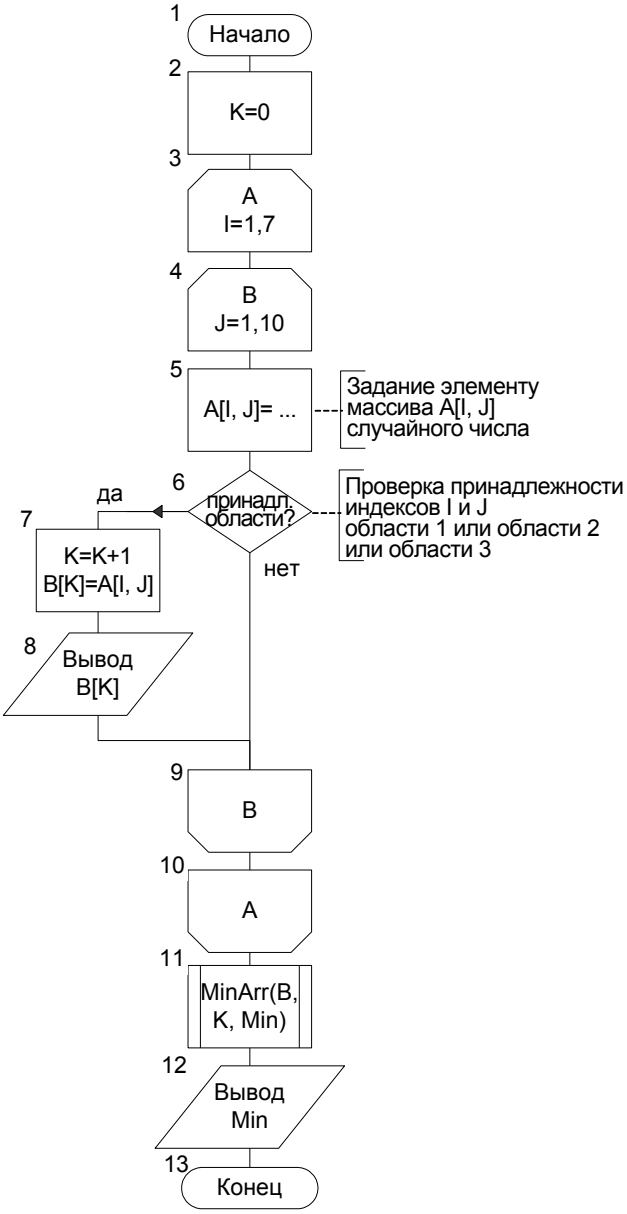

1
Начало
2
K=0
3
A
I=1,7
4
B
J=1,10
5
A[I, J]= ...
Задание элементу
массива A[I, J]
случайного числа
6
принадл.
области?
да
Проверка принадлежности
индексов I и J
области 1 или области 2
или области 3
7
K=K+1
B[K]=A[I, J]
нет
8
Вывод
B[K]
9
B
10
A
11
MinArr(B,
K, Min)
12
Вывод
Min
13
Конец

Рис. 2.8

Задача 2.9. Разработать схему алгоритма и программу для вычисления элементов матрицы A размером 40×50, которые определяются как

$$a_{ij} = \begin{cases} \sin(ij), \text{ если } i \le 10 \text{ и } j \le 10; \\ \cos(i+j), \text{ если } i \ge 30 \text{ и } j \ge 30; \\ \lg^2(i+j) \text{ для остальных } i, j. \end{cases}$$

Найти сумму элементов матрицы, удовлетворяющих условию

$$\begin{cases} \cos^2 a_{ij} > \ln(\sin^2 a_{ij} + 1); \\ a_{ij} > a_{ij}^3. \end{cases}$$

Значение суммы вывести на экран.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.9.

Программа

```
Program Task2_9;
Var
   A  : Array[1..40, 1..50] of Real;
   I, J      : Integer;
   S, R1, R2 : Real;
Begin                {начало раздела операторов программы}
   S:= 0;            {обнуление переменной суммы S}
   for I:=1 to 40 do{цикл 1-для перебора строк матрицы A}
      for J:=1 to 50 do {цикл 2-для перебора столбцов матрицы A}
      begin            {начало циклов 1, 2}
        {вычисление элементов матрицы A}
        If (I<=10) and (J<=10) Then A[I, J]:= Sin(I*J)
        Else If (I>=30) and (J>=30) Then A[I, J]:= Cos(I+J)
             Else A[I, J]:= Ln(I+J)*Ln(I+J)/(Ln(10)*Ln(10));
        R1:=Cos(A[I,J])*Cos(A[I, J]) - Ln(Sin(A[I,J]*Sin(A[I,J])+1));
        R2:=A[I, J]-A[I, J]*A[I, J]*A[I, J];
        {если условие выполняется, то элемент суммируется}
```

```
        If (R1>0) and (R2>0) Then S:= S + A[I, J];
      End;            {конец циклов 1, 2}
   Writeln('Сумма элементов матрицы = ',S:7:3);{вывод суммы на экран}
   Readln;            {ожидание нажатия клавиши Enter}
End.                  {конец программы}
```

Обозначения схемы алгоритма

$A[I, J]$ – a_{ij} элементы матрицы A;

S – значение суммы элементов матрицы a_{ij}, удовлетворяющих заданному условию;

$R1$, $R2$ – значения, необходимые для проверки заданного условия.

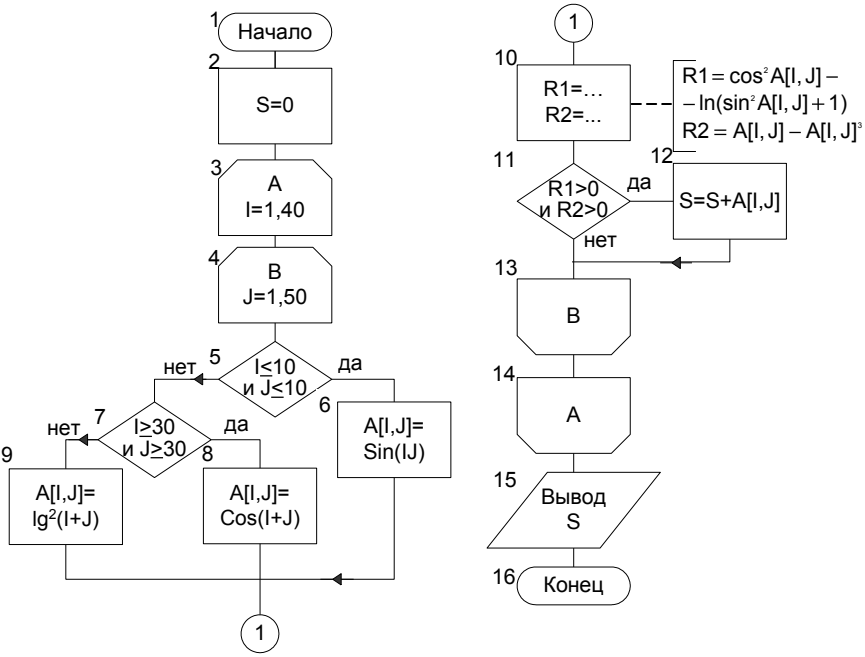


Рис. 2.9

Результат работы программы

```
Сумма элементов матрицы = 52.590
```

***Задача** 2.10*. Разработать схему алгоритма и программу вычисления элементов матрицы A размером 10×8. Элементы матрицы формируются по закону

$$a_{ij} = \begin{cases} \prod_{n=1}^{i} n^2, \text{ если } 1 \le i \le 5,\ 1 \le j \le 4; \\ \sum_{n=1}^{j} \sin(20+n) \text{ для остальных } i \text{ и } j. \end{cases}$$

Вывести значения матрицы на экран построчно.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.10.

Программа

```
Program Task2_10;
Uses Crt;
Var
   I, J, N : Byte;
   A       : Array [1..20, 1..30] of Real;
   S, P    : Real;
Begin              {начало раздела операторов программы}
   ClrScr;         {очистка экрана}
   {формирование матрицы A}
   for I:=1 to 10 do  {цикл 1 - для перебора строк матрицы A}
     for J:=1 to 8 do {цикл 2 - для перебора столбцов матрицы A}
       If (I>=1) and (I<=5) and (J>=1) and (J<=4) Then
       Begin          {начало блока 1 - расчет произведения}
         P:=1;        {первоначальное значение произведения}
         for N := 1 to I do  P := P * N*N;
         A[I, J]:= P;
       End Else       {конец блока 1}
       Begin          {начало блока 2-расчет суммы}
         S:=0;        {первоначальное значение суммы}
```

```
        for N := 1 to J do S := S + Sin(20+N);
        A[I, J]:=S;
      End;          {конец блока 2, конец циклов 2, 1}
      {вывод матрицы на экран}
      for I:=1 to 10 do
      Begin         {начало цикла 3}
        for J:=1 to 8 do Write(A[I, J]:8:1); {вывод строки матрицы A}
        Writeln;  {переход на следующую строку}
      End;          {конец цикла 3}
      Repeat Until KeyPressed;{ожидание нажатия любой клавиши}
End.                {конец программы}
```

Результат работы программы

```
     1.0     1.0     1.0     1.0    -1.1    -0.3     0.7     0.9
     4.0     4.0     4.0     4.0    -1.1    -0.3     0.7     0.9
    36.0    36.0    36.0    36.0    -1.1    -0.3     0.7     0.9
   576.0   576.0   576.0   576.0    -1.1    -0.3     0.7     0.9
 14400.0 14400.0 14400.0 14400.0    -1.1    -0.3     0.7     0.9
     0.8     0.8    -0.0    -0.9    -1.1    -0.3     0.7     0.9
     0.8     0.8    -0.0    -0.9    -1.1    -0.3     0.7     0.9
     0.8     0.8    -0.0    -0.9    -1.1    -0.3     0.7     0.9
     0.8     0.8    -0.0    -0.9    -1.1    -0.3     0.7     0.9
     0.8     0.8    -0.0    -0.9    -1.1    -0.3     0.7     0.9
```

Обозначения схемы алгоритма

$A[I, J]$ – a_{ij} элементы матрицы A;

S – значение суммы выражения $\sum_{n=1}^{j} \sin(20+n)$;

P – значение произведения $\prod_{n=1}^{i} n^2$.

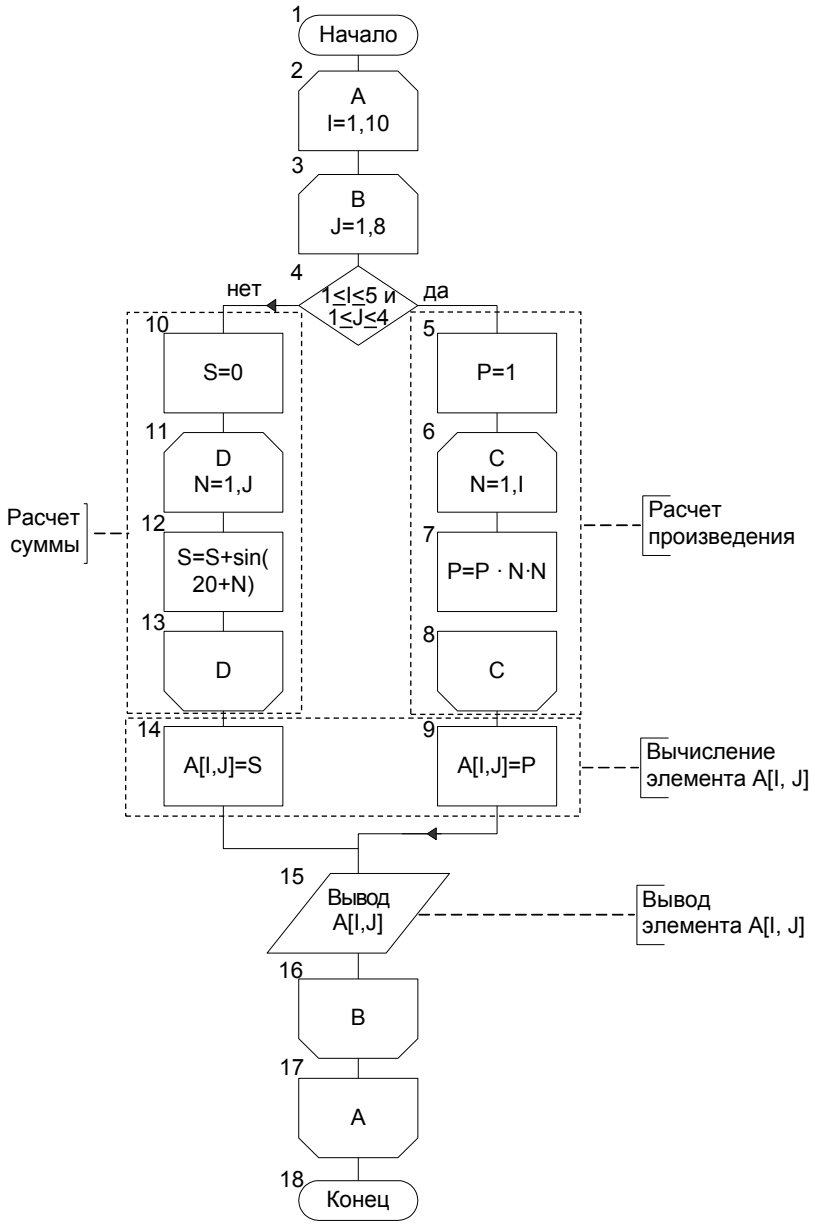

1
Начало
2
A
I=1,10
3
B
J=1,8
4
1≤I≤5 и
1≤J≤4
нет
да
10
S=0
11
D
N=1,J
12
S=S+sin(
20+N)
13
D
14
A[I,J]=S
5
P=1
6
C
N=1,I
7
P=P · N·N
8
C
9
A[I,J]=P
Расчет
суммы
Расчет
произведения
Вычисление
элемента A[I, J]
15
Вывод
A[I,J]
Вывод
элемента A[I, J]
16
B
17
A
18
Конец

Рис. 2.10

Задача 2.11. Разработать схему алгоритма формирования из одномерного массива A массива B, элементы которого удовлетворяют условию $(a_i^2 > \sin a_i)$. Массив A состоит из 200 элементов, которые формируются по закону $a_i = \cos(i)$. Вывести число элементов массива B и его значения на экран построчно, в каждой строке по пять значений.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.11. Построчный вывод элементов массива B реализован с помощью проверки равенства нулю остатка при делении индекса массива B на пять. Если он равен нулю, то осуществляется переход на следующую строку. На языке Pascal проверка этого условия производится с использованием процедуры *mod* (вычисление остатка от деления целых чисел).

Программа

```
Program Task2_11;
Uses Crt;
Var
   A, B    : Array [1..200] of Real;
   I, J, K : Integer;
Begin              {начало раздела операторов программы}
   ClrScr;                 {очистка экрана}
   K := 0;                 {обнуление индекса массива B}
   for I:=1 to 200 do
   Begin                   {начало цикла 1}
     A[I]:=Cos(I);         {вычисление A[I]}
     If A[I]*A[I] > Sin(A[I]) Then
     Else Begin {начало блока 1-если A[I] удовлетв. условию}
            K:= K + 1;     {увеличение индекса K на 1}
            B[K]:= A[I];   {A[I] заносится в массив B}
            Write(B[K]:10:2, ' '); {вывод элемента B[I] на экран}
            {после 5-го элемента- переход на след. строку}
            If K mod 5 = 0 Then   Writeln;
          End;             {конец блока 1}
   End;                    {конец цикла 1}
   Writeln; Writeln('Число элементов = ',K); {вывод числа элементов}
   Repeat Until KeyPressed; {ожидание нажатия любой клавиши}
End.                       {конец программы}
```

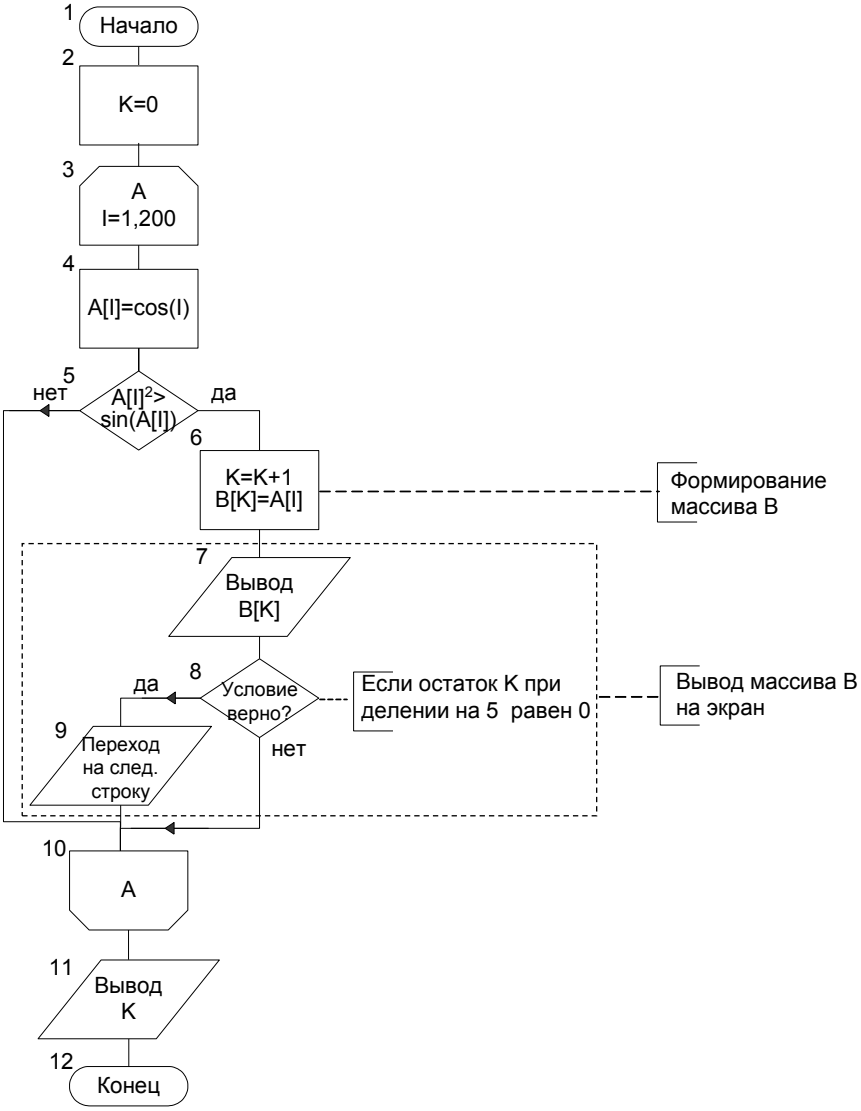

1
Начало
2
K=0
3
A
I=1,200
4
A[I]=cos(I)
5
нет
A[I]²>
sin(A[I])
да
6
K=K+1
B[K]=A[I]
Формирование
массива B
7
Вывод
B[K]
8
да
Условие
верно?
Если остаток K при
делении на 5 равен 0
Вывод массива B
на экран
9
Переход
на след.
строку
нет
10
A
11
Вывод
K
12
Конец

Рис. 2.11

Обозначения схемы алгоритма

$A[I]$ – a_i элемент массива A;
$B[K]$ – b_k элемент массива B;
K – счетчик элементов массива B.

Результат работы программы

```
      0.54      0.28      0.75      0.00      0.84
      0.14      0.66      0.41      0.42      0.65
      0.15      0.83      0.77      0.27      0.56
      0.53      0.30      0.74      0.02      0.85
      0.12      0.67      0.39      0.44      0.63
      0.17      0.82      0.78      0.25      0.57
      0.51      0.32      0.73      0.04      0.86
      0.10      0.69      0.38      0.46      0.62
      0.19      0.81      0.79      0.23      0.58
      0.49      0.33      0.72      0.06      0.87
      0.08      0.70      0.36      0.47      0.61
      0.21      0.80      0.80      0.22      0.60
      0.48      0.35      0.71      0.08      0.88
      0.07      0.71      0.34      0.49
Число элементов = 69
```

***Задача** 2.12.* Разработать схему алгоритма и программу формирования файлов *a.lst, b.lst*, в которые записываются матрицы A, B, полученные по следующим правилам:

$a_{ij} = \cos^2(i \cdot j) \cdot \ln^2(i \cdot j)$, где $i = 1; 3,\ j = 1; 2$ записать в файл a.lst;

$b_{ij} = 3\sin i \cdot \cos j$, где $i = 1; 5,\ j = 1; 4$ записать в файл b.lst.

Все положительные элементы массивов A, B записать в файл *c.lst*.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.12.

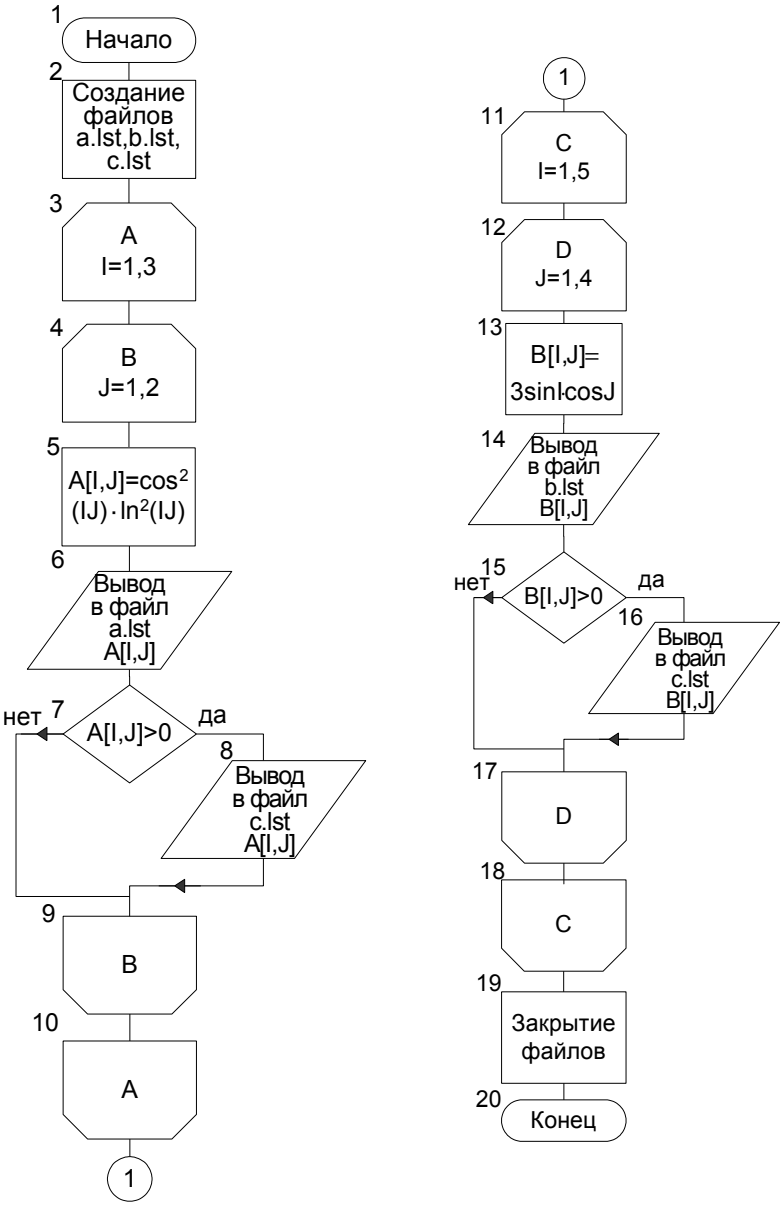

1
Начало
2
Создание файлов a.lst,b.lst, c.lst
3
A
I=1,3
4
B
J=1,2
5
A[I,J]=cos2 (IJ)·ln2(IJ)
6
Вывод в файл a.lst A[I,J]
7
A[I,J]>0
нет
да
8
Вывод в файл c.lst A[I,J]
9
B
10
A
1
1
11
C
I=1,5
12
D
J=1,4
13
B[I,J]= 3sinI·cosJ
14
Вывод в файл b.lst B[I,J]
15
B[I,J]>0
нет
да
16
Вывод в файл c.lst B[I,J]
17
D
18
C
19
Закрытие файлов
20
Конец

Рис. 2.12

Обозначения схемы алгоритма

$A[I, J]$ – a_{ij} элемент матрицы A;
$B[I, J]$ – b_{ij} элемент матрицы B.

Программа

```
Program Task2_12;
Uses Crt;
Var
   F1, F2, F3 : Text;
   I, J : Integer;
   A : Array [1..3, 1..2] of Real;
   B : Array [1..5, 1..4] of Real;
Begin                {начало раздела операторов программы}
   ClrScr;           {очистка экрана}
   {создание и открытие файлов}
   Assign(F1,'a.lst');Assign(F2,'b.lst');Assign(F3,'c.lst');
   Rewrite(F1);  Rewrite(F2); Rewrite(F3);
   {формирование матрицы A}
   for I:=1 to 3 do
   Begin             {начало цикла 1}
     for J:=1 to 2 do
     Begin           {начало цикла 2}
       {вычисление A[I,J]}
       A[I, J] := Cos(I*J)*Cos(I*J) * Ln(I*J)*Ln(I*J);
       {запись элемента A[I, J] в файл a.lst}
       Write(F1, A[I, J]:7:3, ' ');
       {запись положит. элемента A[I, J] в файл c.lst}
       If A[I, J] > 0 Then Write(F3, A[I, J]:7:3, ' ');
     end;            {конец цикла 2}
     Writeln(F1);    {переход на след. строку в файле a.lst}
   end;              {конец цикла 1}
   {формирование матрицы B}
   for I:=1 to 5 do
   Begin             {начало цикла 3}
     for J:=1 to 4 do
     Begin           {начало цикла 4}
       B[I, J]:= 3*Sin(I)*Cos(J);{вычисление B[I, J]}
       {запись элемента B[I, J] в файл b.lst}
       Write(F2, B[I, J]:7:3, ' ');
       {запись положит. элемента B[I, J] в файл c.lst}
       If B[I, J] > 0 Then Write(F3, B[I, J]:7:3, ' ');
     end;            {конец цикла 4}
     Writeln(F2);    {переход на след. строку в файле b.lst}
```

```
   end;              {конец цикла 3}
   Close(F1); Close(F2); Close(F3);{закрытие файлов}
End.                 {конец программы}
```

Результат работы программы

Файл *a.lst*:

```
0.000   0.083
0.083   0.821
1.183   2.960
```

Файл *b.lst*:

```
 1.364  -1.051  -2.499  -1.650
 1.474  -1.135  -2.701  -1.783
 0.229  -0.176  -0.419  -0.277
-1.227   0.945   2.248   1.484
-1.554   1.197   2.848   1.880
```

Файл *c.lst*:

```
  0.083   0.083   0.821   1.183   2.960   1.364   1.474   0.229
0.945   2.248   1.484   1.197   2.848   1.880
```

Задача 2.13. Пусть дано неупорядоченное множество A с числом элементов множества 10 000, сгенерированное случайным образом. Значение каждого элемента является целым числом в диапазоне от 0 до 10 включительно. Разработать схему алгоритма определения числа упорядоченных подмножеств $B = \{2, 5\}$, которые включены в неупорядоченное множество A.

Пояснения к задаче

В приведенном множестве A

$A = \{2, 4, 5, 4, 3, 4, 6, 7, 1, 2, 5, 1, 9, 4, 2, 1, 2, 5, 3, 1, 2, 2, 7, 2, 5\}$ последовательность {2, 5} встречается три раза.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.13.

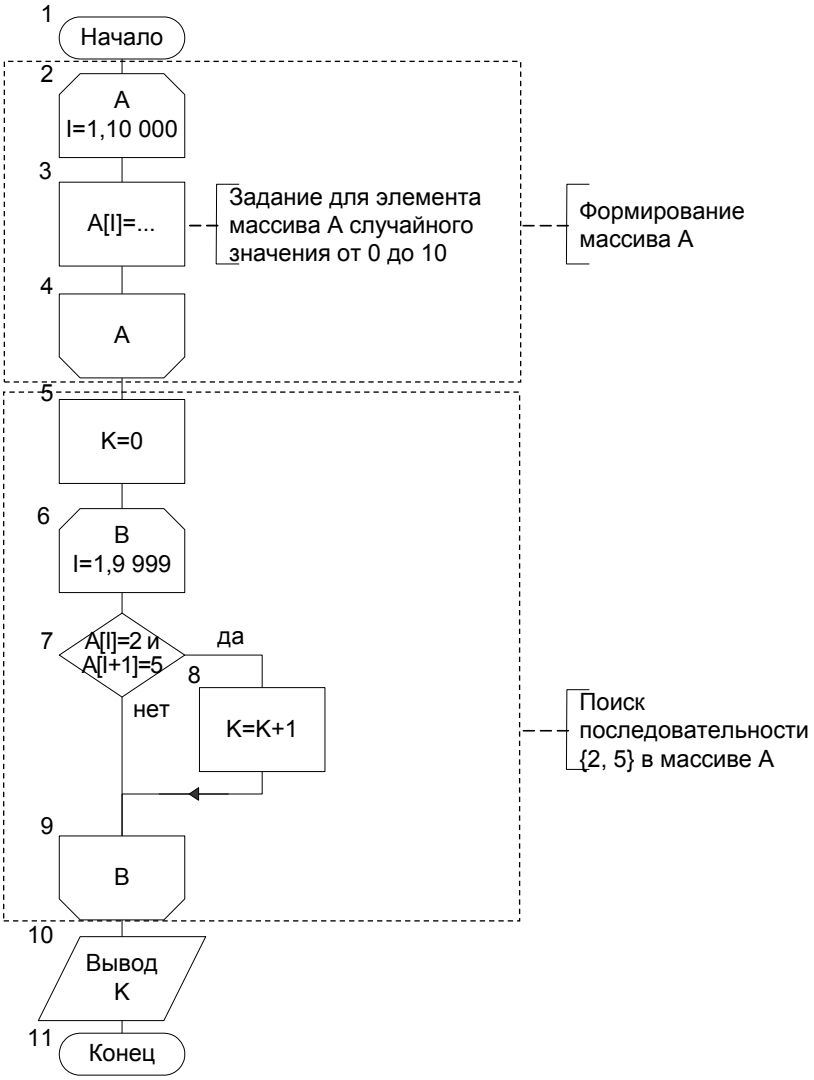

1
Начало
2
A
I=1,10 000
3
A[I]=...
Задание для элемента массива A случайного значения от 0 до 10
Формирование массива A
4
A
5
K=0
6
B
I=1,9 999
7
A[I]=2 и A[I+1]=5
да
8
K=K+1
нет
Поиск последовательности {2, 5} в массиве A
9
B
10
Вывод
K
11
Конец

Рис. 2.13

Обозначения схемы алгоритма

$A[I]$, $A[I+1]$ – a_i, a_{i+1} элементы массива A;
K – число подмножеств $B = \{2, 5\}$.

Программа

```
Program Task2_13;
Var
  A    : Array [1..10000] of Byte;
  I, K : Integer;
Begin         {начало раздела операторов программы}
  for I :=1 to 10000 do  A[I]:=Random(10); {формирование массива A}
  K := 0;
  for I :=1 to 9999 do {цикл – для перебора элементов массива A}
    {проверка является ли последовательность (I,I+1) заданной (2,5)}
    If (A[I] = 2) and (A[I + 1] = 5) then K := K + 1;
  Writeln('Число последовательностей = ',K); {вывод результата}
End.          {конец программы}
```

Результат работы программы

```
Число последовательностей = 92.
```

2.4. **Задачи по вычислению функциональных выражений с использованием сложных сумм и произведений**

Задача 2.14. Построить схему алгоритма и программу вычисления и вывода на экран значения суммы $S = \sum_{i=1}^{10} \sum_{j=1}^{20} \sum_{k=1}^{30} e^{-i-j-k}$.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.14.

Программа

```
Program Task2_14;
Var
   I, J, K : Byte;
   S       : Real;
```

```
Begin          {начало раздела операторов программы}
   S:= 0;      {обнуление переменной суммы S}
   for I:=1 to 10 do
     for J:=1 to 20 do
       for K:=1 to 30 do
         S := S + Exp(-I-J-K);
   Writeln('Сумма = ', S:6:3);  {вывод суммы на экран}
   Readln;     {ожидание нажатия клавиши Enter}
End.           {конец программы}
```

Результат работы программы

```
Сумма =  0.197
```

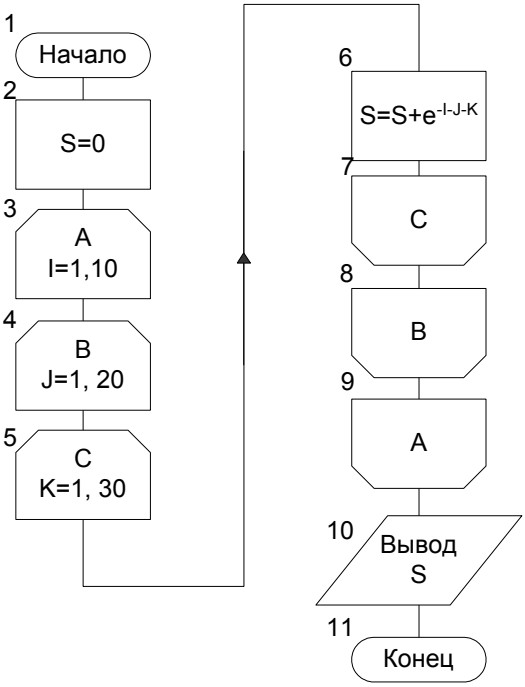


Рис. 2.14

Задача 2.15. Разработать схему алгоритма и программу вычисления сумм для каждого элемента массива $X = \{4{,}1;\ 5{,}3;\ 6{,}1;\ 7{,}2;\ 10{,}1\}$ по формуле $S(x_i) = \log_2(3x_i + 1) + \log_3(4x_i + 1) + \log_4(5x_i + 1) + \ldots + \log_{101}(102x_i + 1)$. Вывести значения сумм на экран.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.15.

Программа

```
Program Task2_15;
Var
   I, J : Byte;
   X : Array[1..5] of Real;
   S : Real;
Begin  {начало раздела операторов программы}
   Write('Введите 5 значений: ');
   for I:=1 to 5 do Read(X[I]); {ввод значений массива X}
     for I:=1 to 5 do
     Begin    {начало цикла 1 – для перебора значений массива X}
       S:=0;  {первоначальное значение переменной суммы S}
       {суммирование}
       For J:=2 to 101 do S := S + Ln((J+1)*X[I]+1)/Ln(J);
       Writeln('X= ',X[I]:5:1,' S= ',S:7:2);{вывод результатов на экран}
     End;            {конец цикла 1}
   Readln;           {ожидание нажатия клавиши Enter}
End.                 {конец программы}
```

Результат работы программы

```
Введите 5 значений: 4.1 5.3 6.1 7.2 10.1
X=   4.1 S=  145.16
X=   5.3 S=  152.81
X=   6.1 S=  157.01
X=   7.2 S=  161.98
X=  10.1 S=  172.13
```

Обозначение схемы алгоритма

$X[I]$ – x_i элемент массива X.

Обозначения схемы алгоритма

$S1$, $S2$ – значения предыдущего (S_n) и следующего (S_{n+1}) элементов ряда;

Sum – значение суммы членов ряда.

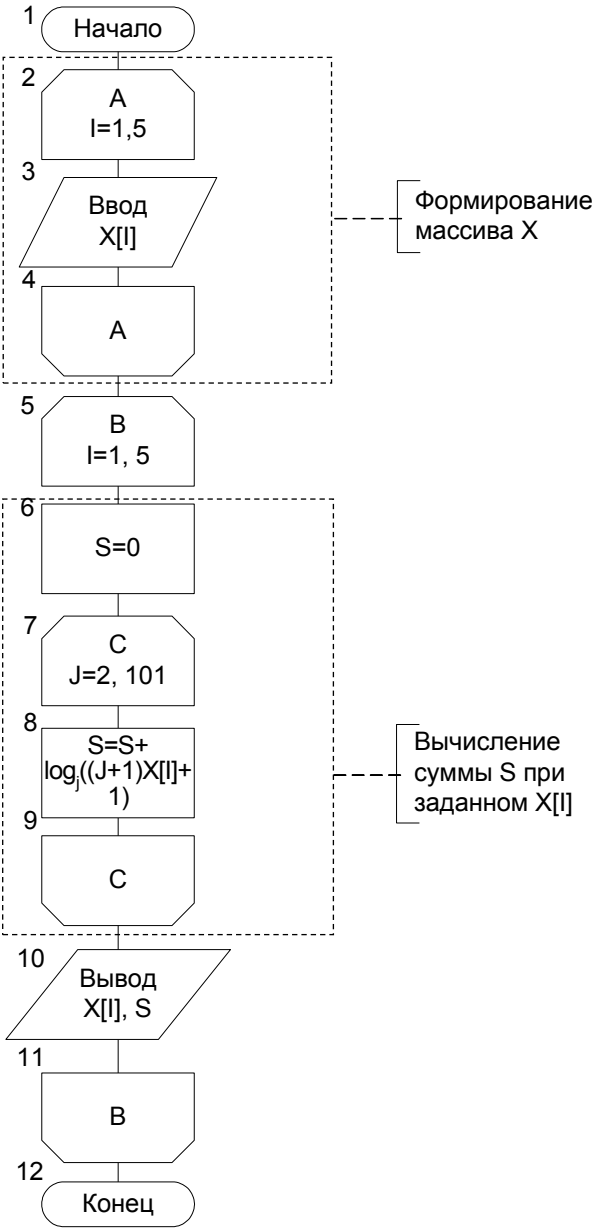

1
Начало
2
A
I=1,5
3
Ввод
X[I]
4
A
Формирование
массива X
5
B
I=1, 5
6
S=0
7
C
J=2, 101
8
S=S+
log_j((J+1)X[I]+
1)
9
C
Вычисление
суммы S при
заданном X[I]
10
Вывод
X[I], S
11
B
12
Конец

Рис. 2.15

Задача 2.16. Разработать схему алгоритма и программу вычисления суммы членов ряда $S = \sum_{x=1}^{\infty} \frac{-1{,}5^x}{\ln(\sin x)}$. Ограничиться числом членов ряда, для которых выполняется условие: разница между соседними членами ряда меньше 0,001. Если $\sin x \le 0$, то положить $\ln(\sin x) = x^5$. Значение суммы ряда вывести на экран.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.16.

Программа

```
Program Task2_16;
Uses Crt;
Var
  X : Integer;
  Sum, S1, S2 : Real;
Begin                {начало раздела операторов программы}
  ClrScr;            {очистка экрана}
  X:= 1;             {начальное значение переменной X}
  S2 := -1.5/(Ln(Sin(X)));
  Sum:= S2;
  Repeat             {начало цикла 1}
    S1 := S2;        {предыдущий элемент ряда S1}
    X := X + 1;      {приращение переменной X}
    {вычисление следующего элемента ряда S2}
    If Sin(X) > 0 Then S2 := - Exp(X*Ln(1.5))/Ln(Sin(X))
    Else S2 := - Exp(X*Ln(1.5))/Exp(5*Ln(X));
    Sum:= Sum + S2; {подсчет суммы ряда}
  Until Abs(S1 - S2) < 0.001;    {конец цикла 1}
  Write('Сумма ряда = ',Sum:6:2); {вывод результата на экран}
  Repeat Until KeyPressed; {ожидание нажатия любой клавиши}
End.                         {конец программы}
```

Результат работы программы

```
Сумма ряда = 34.07
```

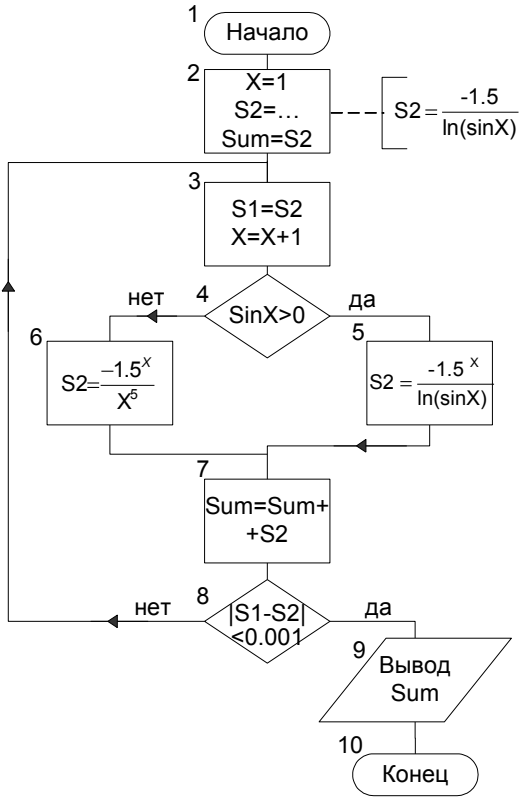


Рис. 2.16

***Задача** 2.17.* Разработать схему алгоритма и программу решения квадратного уравнения $ax^2 + bx + c = 0$ с параметрами a, b, c, которые вычисляются по формулам: $a = \dfrac{1}{\sin R + 1{,}5} - \sum_{n=R}^{20} \dfrac{1}{R^n}$;

$b = e^{R/50} + \prod_{n=1}^{30} \sin n$; $c = a \cdot b + \ln|a| \cdot \ln|b|$; R = 10, 100 (5). В случае, если есть решение ($D = b^2 - 4a \cdot c \geq 0$), вывести на экран корни урав-

нения и значение *R*, если решения нет, то вывести сообщение «Действительных корней нет» и значения *R*, *a*, *b*, *c*.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.17.

Программа

```
Program Task2_17;
Var
   A, B, C, X1, X2, D : Real;
   N, R               : Integer;
Begin               {начало раздела операторов программы}
   R := 10;         {начальное значение параметра цикла 1}
   While R <= 100 do
   Begin            {начало цикла 1}
     {вычисление параметра A}
     A := 1/(Sin(R)+1.5);
     for N := R to 20 do   A := A - 1/Exp(N*Ln(R));
     {вычисление параметра B}
     B := 1;
     for N := 1 to 30 do   B := B*Sin(N);
     B := Exp(R/50) + B;
     C := A*B + Ln(Abs(A)) * Ln(Abs(B));{вычисление параметра C}
     D := B*B - 4*A*C;     {вычисление дискриминанта}
     If D >= 0 Then {начало блока 1 – вычисление корней}
     Begin
       X1 := (-B + Sqrt(D))/(2*A);  X2:= (-B - Sqrt(D))/(2*A);
       {вывод значений корней на экран}
       Writeln(' R= ',R:3,' X1 = ',X1:5:2,' X2 = ', X2:5:2);
     end            {конец блока 1}
     Else Writeln(' R= ', R:3, ' Действительных корней нет. A=
', A:5:2, ' B= ', B:5:2, ' C= ', C:5:2);
     R := R + 5;    {приращение параметра цикла 1}
   end;             {конец цикла 1}
   Readln;          {ожидание нажатия клавиши Enter}
end.                {конец программы}
```

Обозначения схемы алгоритма

*X*1, *X*2 – корни квадратного уравнения.

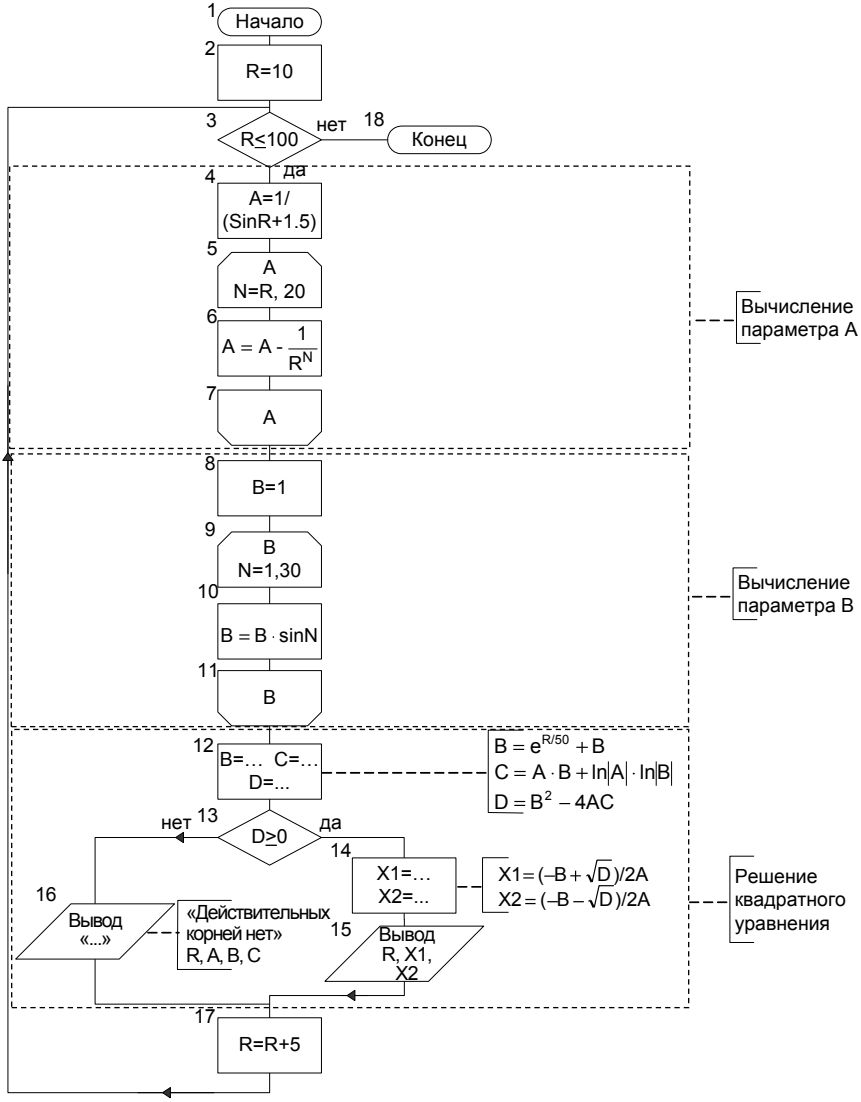

1
Начало
2
R=10
3
R≤100
нет
18
Конец
да
4
A=1/
(SinR+1.5)
5
A
N=R, 20
6
A = A - 1/R^N
7
A
Вычисление параметра A
8
B=1
9
B
N=1,30
10
B = B · sinN
11
B
Вычисление параметра B
12
B=… C=…
D=...
B = e^(R/50) + B
C = A · B + ln|A| · ln|B|
D = B^2 − 4AC
нет
13
D≥0
да
14
X1=…
X2=...
X1 = (–B + √D)/2A
X2 = (–B – √D)/2A
Решение квадратного уравнения
16
Вывод
«...»
«Действительных корней нет»
R, A, B, C
15
Вывод
R, X1,
X2
17
R=R+5

Рис. 2.17

Результат работы программы

```
R=  10 Действительных корней нет. A= 1.05 B=  1.22 C=  1.29
R=  15 X1 = -0.33 X2 = -2.57
R=  20 X1 = -0.19 X2 = -3.41
R=  25 Действительных корней нет. A=  0.73 B=  1.65 C=  1.05
R=  30 Действительных корней нет. A=  1.95 B=  1.82 C=  3.96
R=  35 Действительных корней нет. A=  0.93 B=  2.01 C=  1.83
R=  40 X1 = -0.16 X2 = -4.84
R=  45 X1 = -0.11 X2 = -5.67
R=  50 X1 = -1.07 X2 = -2.29
R=  55 Действительных корней нет. A=  2.00 B=  3.00 C=  6.77
R=  60 X1 = -1.05 X2 = -2.92
R=  65 X1 = -0.13 X2 = -8.41
R=  70 X1 = -0.16 X2 = -9.06
R=  75 X1 = -1.11 X2 = -3.87
R=  80 Действительных корней нет. A=  1.98 B=  4.95 C= 10.88
R=  85 X1 = -0.74 X2 = -6.50
R=  90 X1 = -0.16 X2 = -14.32
R=  95 X1 = -0.24 X2 = -14.36
R= 100 X1 = -1.21 X2 = -6.14
```

2.5. **Задача по вычислению функции, заданной в интервале**

Задача 2.18. Разработать схему алгоритма и программу для вычисления интервальной функции z для десяти значений x_i, заданных массивом, которые вводятся с клавиатуры:

$$z=\begin{cases} 10x_i^2+3b, & \text{если } x_i<100; \\ \sin\left[x_i^{3/5}\cos(ax_i)\right], & \text{если } 100\le x_i\le 300; \\ 11x_i+x_i^a, & \text{если } x_i>300, \end{cases}$$

где a и b – параметры, рассчитываемые как

$$a=\sum_{j=1}^{30}\cos(j+3),\ b=1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{100}.$$

На экран выводить значения интервальной функции z для каждого x_i.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.18.

Программа

```
Program Task2_18;
Uses Crt;
Var
  X         : Array[1..10] of Real;
  z, a, b   : Real;
  I         : Integer;
Begin               {начало раздела операторов программы}
  ClrScr;           {очистка экрана}
  Write('Введите массив из 10 чисел: ');
  for I:=1 to 10 do  Read(X[I]);  {ввод значений массива X}
  a:=0;   b:=1;
  for I:=1 to 30 do  a := a + Cos(I+3);{вычисление параметра a}
  for I:=2 to 100 do b := b + 1/i; {вычисление параметра b}
  {для X[I] вычисление значения функции Z}
  for I := 1 to 10 do
  Begin                     {начало цикла}
    If X[I] < 100 Then Z:= 10*X[I]*X[I] + 3*b
         Else If X[I]>300 Then Z := 11*X[I]+Exp(a*Ln(X[I]))
                          Else Z:= Sin(Exp(3/5*Ln(X[I]))*Cos(a*X[I]));
    Writeln('Z', I,'=', Z:10:3);{вывод на экран результатов}
  end;                      {конец цикла}
  Repeat Until KeyPressed;{ожидание нажатия любой клавиши}
End.                        {конец программы}
```

Результат работы программы

```
Введите массив из 10 чисел: 1 150 250 350 10000 -235 0.02 -0.005 789 10
Z1=    25.562
Z2=     0.811
Z3=    -0.973
Z4=  5596.076
Z5=235154.762
Z6=552265.562
Z7=    15.566
Z8=    15.562
Z9=  13598.539
Z10=  1015.562
```

Обозначение схемы алгоритма

$X[I]$ – x_i элемент массива X.

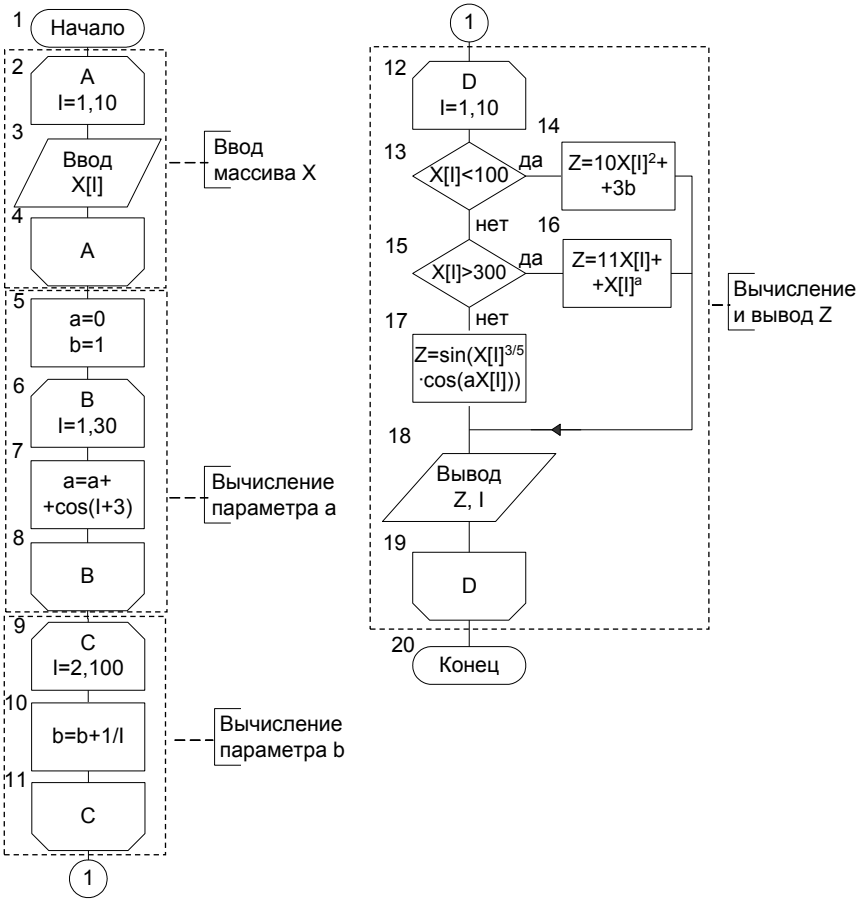


Рис. 2.18

2.6. **Задачи по нахождению суммы конечного и бесконечного функциональных рядов**

Задача 2.19. Разработать схему алгоритма и программу вычисления ряда $S(x) = 2\sin x + 4\sin^3 x + 6\sin^5 x + ... + 20\sin^{19} x$, если

переменная *X* может изменяться по закону *X* = 1; 3 (0,5). Вывести на экран значения сумм *S* для каждого значения *X*.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.19.

Программа

```
Program Task2_19;
Uses Crt;
Var
   J    : Byte;
   P, S, X : Real;
Begin                {начало раздела операторов программы}
   ClrScr;           {очистка экрана}
   X:= 1;            {начальное значение переменной X}
   While X <= 3 do
   Begin             {начало цикла 1 - для перебора значений X}
     S:= 0;          {начальное значение переменной суммы S}
     P := sin(X);
     for J := 1 to 10 do {цикл для суммирования членов ряда}
     Begin
       S := S + 2*J * P; {вычисление суммы}
       {вычисление произведения для элементов ряда}
       P := P * sin(X)*sin(X);
     End;
     Writeln('X = ',X:4:1,' S = ',S:6:2);{вывод результатов}
     X:= X + 0.5;    {приращение переменной X}
   end;              {конец цикла 1}
   Repeat Until KeyPressed;{ожидание нажатия любой клавиши}
End.                 {конец программы}
```

Результат работы программы

```
X = 1.0 S =   17.30
X = 1.5 S =  106.48
X = 2.0 S =   35.90
X = 2.5 S =    2.90
X = 3.0 S =    0.29
```

Обозначение схемы алгоритма

P – промежуточная переменная для вычисления значений элементов ряда.

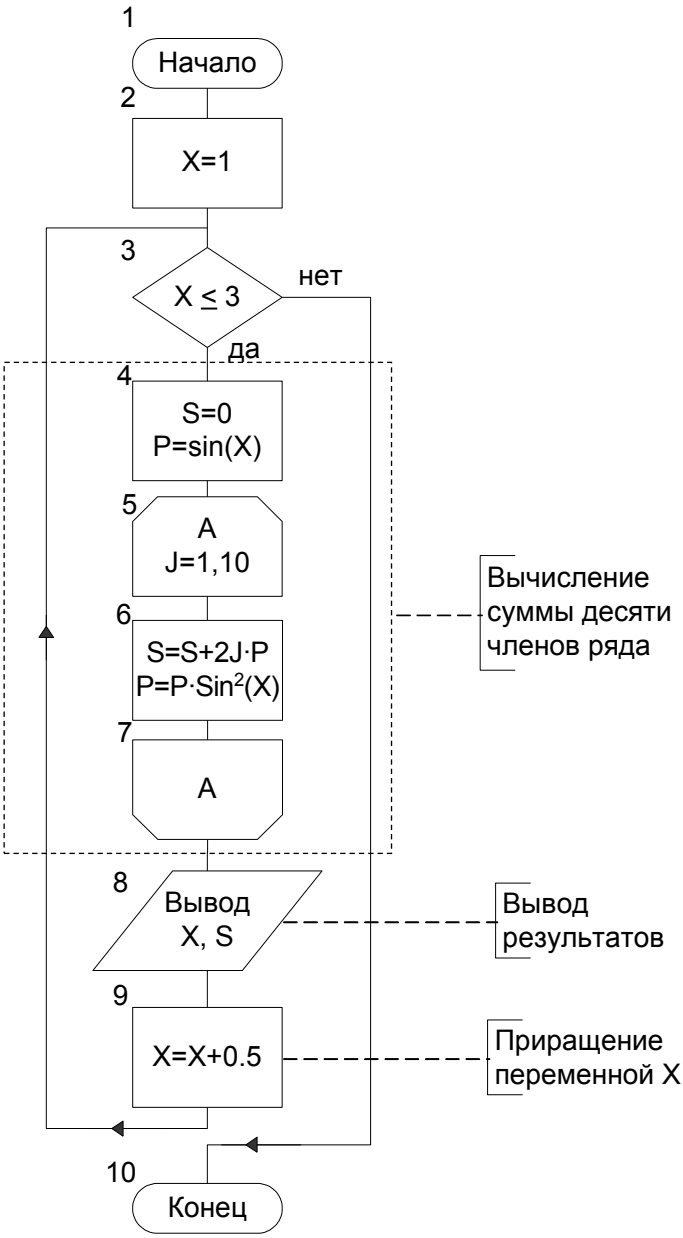

1
Начало
2
X=1
3
X ≤ 3
нет
да
4
S=0
P=sin(X)
5
A
J=1,10
6
S=S+2J·P
P=P·Sin²(X)
7
A
Вычисление суммы десяти членов ряда
8
Вывод
X, S
Вывод результатов
9
X=X+0.5
Приращение переменной X
10
Конец

Рис. 2.19

Задача 2.20. Разработать схему алгоритма и программу вычисления суммы членов ряда $S = \sum_{x=1}^{\infty}(\frac{1}{\ln^9(x+2)} + \frac{1}{x})$. Ограничиться числом членов ряда, для которых выполняется условие $|S_n - S_{n-1}| < 0{,}01$, где $S_n = \sum_{x=1}^{n}(\frac{1}{\ln^9(x+2)} + \frac{1}{x})$ – сумма n членов ряда. Использовать функцию для вычисления члена ряда.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.20.

Программа

```
Program Task2_20;
Var
   X : Integer;
   S1, S2, E : Real;
{функция вычисления члена ряда}
Function F(x :Integer) : Real;
Begin           {начало раздела операторов функции F}
   F:=1/Exp(9*Ln(x+2))+1/x;
End;            {конец функции F}

Begin  {начало раздела операторов основной программы}
   X:= 1;
   S2:= F(X);    {сумма, состоящая из 1-го члена ряда}
   Repeat        {начало цикла}
      S1:= S2;  {S1-сумма, состоящая из (X-1) членов ряда}
      X:= X + 1;      {увеличение X на 1}
      S2:= S2 + F(X);{S2-сумма, состоящая из X членов ряда}
      E:= Abs(S2 - S1);{вычисление погрешности E}
   Until E < 0.01; {конец цикла-пока погрешность не станет < 0.01}
   Writeln('Сумма ряда = ', S2:5:3); {вывод результата на экран}
   Readln;  {ожидание нажатия клавиши Enter}
End.         {конец основной программы}
```

Результат работы программы

```
Сумма ряда =  5.197
```

Обозначения схемы алгоритма

S1 – значение суммы, состоящей из (*X* – 1) членов ряда;

S2 – значение суммы, состоящей из *X* членов ряда;

F(X) – вызов функции *F* для вычисления значения члена ряда в зависимости от значения *X*;

E – значение погрешности вычислений.

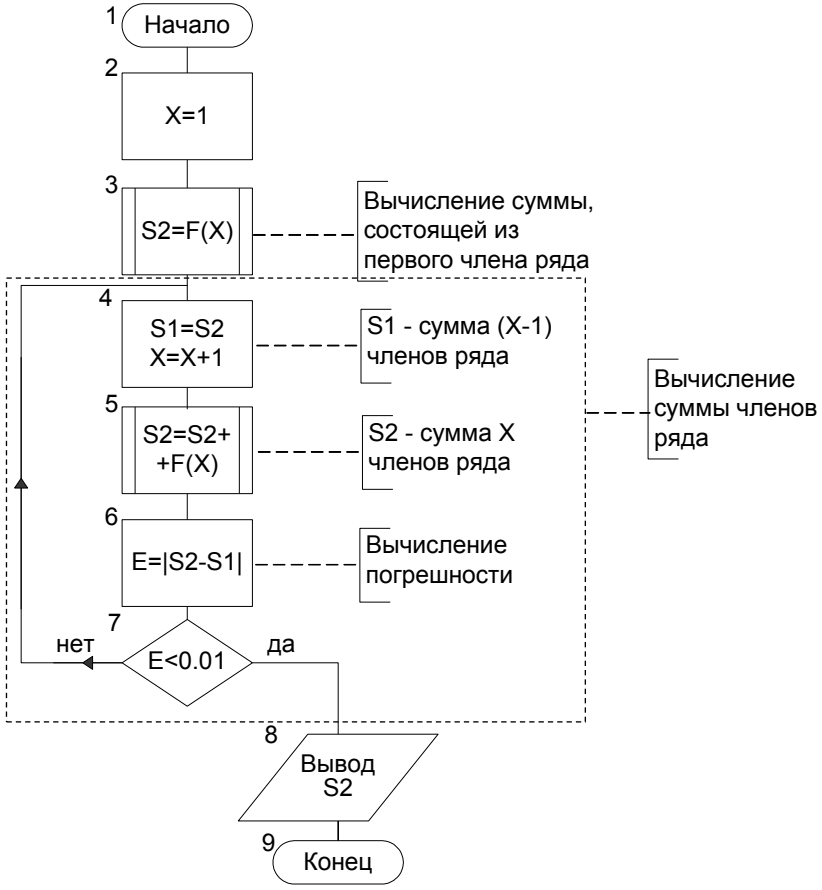


Рис. 2.20

2.7. **Задачи с использованием двойных сумм и произведений**

Задача 2.21. Разработать схему алгоритма и программу формирования матрицы A размером 20×20 по закону

$$a_{ij} = \begin{cases} \sum\limits_{n=1}^{i} \sum\limits_{m=1}^{j} 2\sin(n+m), \text{если } i > j; \\ \prod\limits_{n=1}^{i} \prod\limits_{m=1}^{j} 2\sin(n+m), \text{если } i \le j. \end{cases}$$

Пояснения:

$$a_{21} = \sum_{n=1}^{2} \sum_{m=1}^{1} 2\sin(n+m) = \sum_{n=1}^{2} 2\sin(n+1) =$$
$$= 2\sin(1+1) + 2\sin(2+1), \text{ так как } i > j;$$

$$a_{32} = \sum_{n=1}^{3} \sum_{m=1}^{2} 2\sin(n+m) = \sum_{n=1}^{3} \left[\sin(n+1) + 2\sin(n+2)\right] =$$
$$2\sin(1+1) + 2\sin(1+2) + 2\sin(2+1) + 2\sin(2+2) +$$
$$+ 2\sin(3+1) + 2\sin(3+2), \text{ так как } i > j;$$

$$a_{23} = \prod_{n=1}^{2} \prod_{m=1}^{3} 2\sin(n+m) = \prod_{n=1}^{2} 2\sin(n+1) \cdot 2\sin(n+2) \cdot 2\sin(n+3) =$$
$$= 2\sin(1+1) \cdot 2\sin(1+2) \cdot 2\sin(1+3) \cdot 2\sin(2+1) \cdot 2\sin(2+2) \cdot 2\sin(2+3),$$
$$\text{так как } i \le j.$$

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.21.

Обозначения схемы алгоритма

S – значение двойной суммы, вычисляемое по формуле $\sum_{n=1}^{i}\sum_{m=1}^{j} 2\sin(n+m)$;

P – значение двойного произведения, вычисляемое по формуле $\prod_{n=1}^{i}\prod_{m=1}^{j} 2\sin(n+m)$.

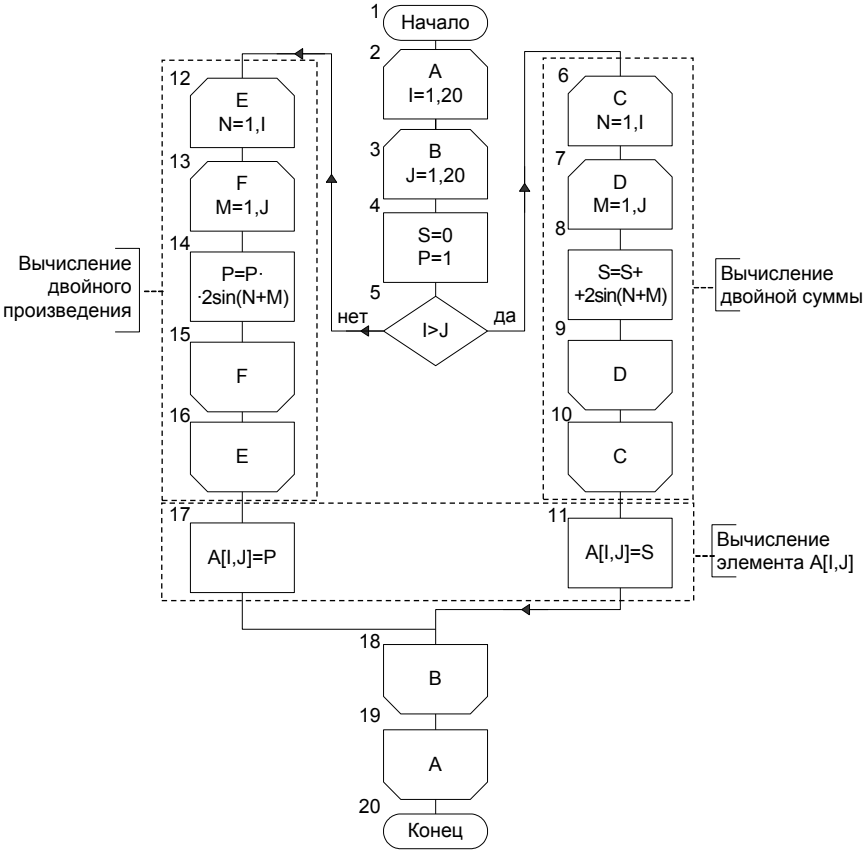


Рис. 2.21

Программа

```
Program Task2_21;
Var
   A : Array[1..20, 1..20] of Real;
   I, J, N, M : Byte;
   S, P       : Real;
Begin    {начало раздела операторов программы}
   for I := 1 to 20 do  {цикл 1-для перебора строк матрицы A}
     for J := 1 to 20 do {цикл 2-для перебора столбцов матрицы A}
     Begin              {начало циклов 1, 2}
       P:=1;  S:=0; {начальные значения для произведения и суммы}
       If I > J Then
       Begin                    {начало блока 1, если I > J}
         {вычисление двойной суммы}
         for N := 1 to I do
           for M := 1 to J do  S:= S + 2*Sin(N+M);{суммирование}
         A[I, J] := S;
       End Else                 {конец блока 1}
           Begin                {начало блока 2, если I ≤ J }
             {вычисление двойного произведения}
             for N := 1 to I do
               for M := 1 to J do     P := P * 2*Sin(N+M);
             A[I, J] := P;
           end;                 {конец блока 2}
     end;                       {конец цикла 2, 1}
End.                            {конец программы}
```

2.8. **Задача с использованием процедур и функций для двумерного массива**

Задача 2.22. Разработать схему алгоритма и программу формирования матрицы A размером 20×40, элементы которой формируются по закону

$$a_{ij} = \begin{cases} \left|\cos(i+j)+1\right|^{3,4}, & \text{если } i = j; \\ \prod\limits_{k=1}^{10} \sin(i+j)\cdot k, & \text{если } i < j; \\ \sum\limits_{k=11}^{30} \ln(i+j)\cdot(-k), & \text{если } i > j. \end{cases}$$

В программе создать процедуру, которая считает количество положительных и отрицательных элементов. Результаты расчета выводятся на экран.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.22, *а*. Схема алгоритма процедуры подсчета положительных и отрицательных элементов *Calc* представлена на рис. 2.22, *б*.

Программа

```
Program Task2_22;
Uses Crt;
Type
   Mas = Array [1..20, 1..40] of Real;
Var
   A : Mas;
   I, J, K, CP, CN : Integer;
   S,P : Real;
{Процедура подсчета положительных (K1) и отрицательных (K2)
элементов массива B}
Procedure Calc(B :Mas; Var K1, K2 :Integer);
Var  I, J : Integer;
Begin         {начало раздела операторов процедуры Calc}
   K1 := 0;      K2 := 0;
   for I := 1 to 20 do
     for J := 1 to 40 do
       If B[I, J] > 0 Then K1 := K1+1 {если B[I,J] положительный}
       {если B[I,J] отрицательный}
       Else If B[I, J] < 0 Then K2 := K2 + 1;
end;                 {конец процедуры Calc}

Begin         {начало раздела операторов основной программы}
    ClrScr;             {очистка экрана}
    for I:=1 to 20 do  {цикл 1-для перебора строк матрицы A}
     for J:=1 to 40 do {цикл 2-для перебора столбцов матрицы A}
     Begin             {начало цикла 1, 2}
         If I = J
           Then A[I, J] := Exp(3.4 * Ln(Abs(Cos(I+J)+1)))
           Else If I > J Then
                Begin    {начало блока 1-суммирование}
                  S := 0;
                  for K := 11 to 30 do S := S + Ln(I+J)*(-K);
```

```
                    A[I, J] := S;
                  End          {конец блока 1}
                  Else Begin {начало блока 2-произведение}
                         P := 1;
                         for K := 1 to 10 do P := P * Sin(I+J)*K;
                         A[I, J] := P;
                       end; {конец блока 2}
     end;                    {конец циклов 2, 1}
    Calc(A, CP, CN);        {вызов процедуры Calc}
    {вывод результатов расчета на экран}
    Writeln('Положительных элементов в матрице = ',CP);
    Writeln('Отрицательных элементов в матрице = ',CN);
    Repeat Until KeyPressed;{ожидание нажатия любой клавиши}
End.                         {конец основной программы}
```

Результат работы программы

```
Положительных элементов в матрице = 610
Отрицательных элементов в матрице = 190
```

Обозначения схемы алгоритма (рис. 2.22, а)

$A[I, J]$ – a_{ij} элемент матрицы A;

S – значение суммы выражения $\sum_{k=11}^{30} \ln(i+j)\cdot(-k)$;

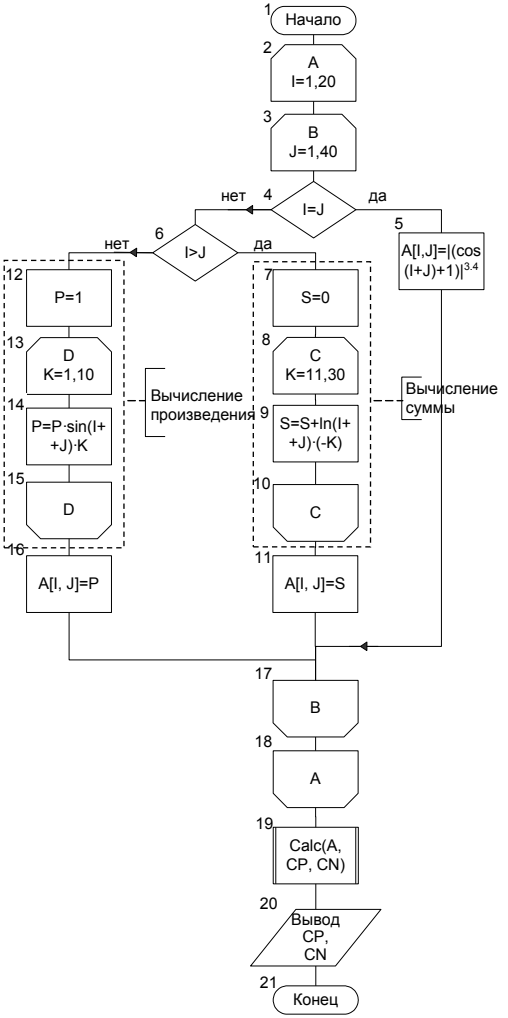
P – значение произведения $\prod_{k=1}^{10} \sin(i+j)k$;

CP, CN – число положительных и число отрицательных элементов матрицы A.

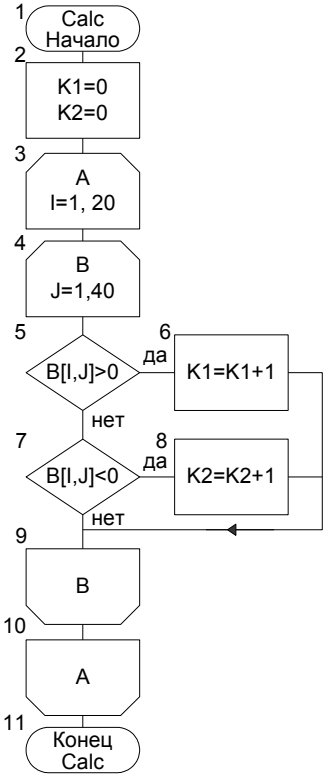
Обозначения схемы алгоритма процедуры Calc (см. рис. 2.22, б)

B – массив, переданный в процедуру в качестве формального параметра;

$K1$, $K2$ – число положительных и число отрицательных элементов массива B.


1
Начало
2
A
I=1,20
3
B
J=1,40
4
I=J
нет
да
5
A[I,J]=|(cos
(I+J)+1)|3,4
6
I>J
нет
да
12
P=1
13
D
K=1,10
14
P=P·sin(I+
+J)·K
15
D
16
A[I, J]=P
7
S=0
8
C
K=11,30
9
S=S+ln(I+
+J)·(-K)
10
C
11
A[I, J]=S
Вычисление
произведения
Вычисление
суммы
17
B
18
A
19
Calc(A,
CP, CN)
20
Вывод
CP,
CN
21
Конец

а


1
Calc
Начало
2
K1=0
K2=0
3
A
I=1, 20
4
B
J=1,40
5
B[I,J]>0
6
да
K1=K1+1
нет
7
B[I,J]<0
8
да
K2=K2+1
нет
9
B
10
A
11
Конец
Calc

б

Рис. 2.22

2.9. **Задача с элементами линейного программирования**

Задача 2.23. Разработать схему алгоритма и программу для вычисления максимального значения функции $F = 3x_1 + 4x_2 + 5x_1x_2x_3$ при дискретных значениях аргументов: x_1 = 1; 100 (1), x_2 = 10; 50 (0,4), x_3 = 5; 30 (0,3). Область определения функции F задана ограничениями

$$\begin{cases} 4x_1 + 5x_2 - 3x_3 \le Q_1, \\ 5x_1 - x_2 + x_1x_3 \le Q_2, \\ -6x_1 - 2x_2x_3 \le Q_3. \end{cases}$$

Значения ограничений Q_1, Q_2, Q_3 вводятся с клавиатуры. Вычисление функции F оформить в виде подпрограммы (*function*). Вывести на экран максимум функции F и соответствующие ей значения переменных x_1, x_2, x_3. Если при заданных ограничениях отсутствует область значений функции F, то вывести соответствующее сообщение.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.23.

Обозначения схемы алгоритма

MaxF – максимальное значение функции F;
ValF – значение функции F;
X1Max, *X2Max*, *X3Max* – значения переменных x_1, x_2, x_3, при которых функция F принимает максимальное значение.

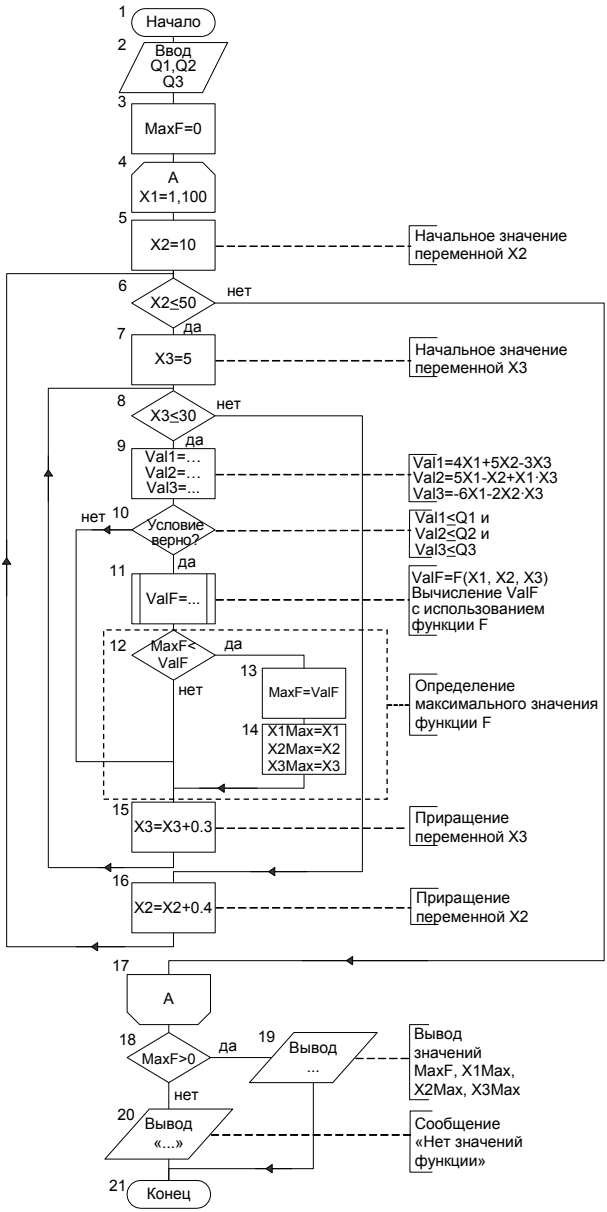

1
Начало
2
Ввод
Q1,Q2
Q3
3
MaxF=0
4
A
X1=1,100
5
X2=10
Начальное значение переменной X2
6
X2≤50
нет
да
7
X3=5
Начальное значение переменной X3
8
X3≤30
нет
да
9
Val1=…
Val2=…
Val3=...
Val1=4X1+5X2-3X3
Val2=5X1-X2+X1·X3
Val3=-6X1-2X2·X3
нет
10
Условие верно?
Val1≤Q1 и
Val2≤Q2 и
Val3≤Q3
да
11
ValF=...
ValF=F(X1, X2, X3)
Вычисление ValF
с использованием
функции F
12
MaxF< ValF
да
нет
13
MaxF=ValF
14
X1Max=X1
X2Max=X2
X3Max=X3
Определение максимального значения функции F
15
X3=X3+0.3
Приращение переменной X3
16
X2=X2+0.4
Приращение переменной X2
17
A
18
MaxF>0
да
19
Вывод
...
Вывод значений MaxF, X1Max, X2Max, X3Max
нет
20
Вывод
«...»
Сообщение «Нет значений функции»
21
Конец

Рис. 2.23

Программа

```
Program Task2_23;
Uses Crt;
Var
   Q1, Q2, Q3 : Real;
   X1, X1Max  : Integer;
   Val1, Val2, Val3, X2, X3 : Real;
   ValF, MaxF, X2Max, X3Max : Real;
{функция вычисления значения заданной функции}
Function F(X1 :Integer; X2, X3 :Real) : Real;
Begin              {начало раздела операторов функции F}
   F:=3*X1 + 4*X2 + 5*X1*X2*X3;
End;          {конец функции F}
Begin         {начало раздела операторов основной программы}
   ClrScr;    {очистка экрана}
   Write('Введите значения ограничений Q1, Q2, Q3: ');
   Read(Q1, Q2, Q3); {ввод значений ограничений}
   MaxF:= 0;          {начальное значение максимума функции}
   For X1 := 1 to 100 do
   Begin      {начало цикла 1 - для перебора значений X1}
     X2:= 10;       {начальное значение переменной X2}
     While X2 <= 50 do
     Begin    {начало цикла 2 - для перебора значений X2}
       X3:= 5; {начальное значение переменной X3}
       While X3 <= 30 do
       Begin {начало цикла 3 - для перебора значений X3}
         {вычисление значений для выполнения ограничений}
         Val1:= 4*X1 + 5*X2 - 3*X3;
         Val2:= 5*X1 - X2 + X1*X3;  Val3:= -6*X1 - 2*X2*X3;
         {проверка выполнения ограничений}
         If (Val1<=Q1) and (Val2<=Q2) and (Val3<=Q3) Then
         Begin  {начало блока 1 - если ограничения выполняются}
           ValF:= F(X1, X2, X3);{вычисление значения функции}
           {поиск максимального значения}
           If MaxF < ValF Then
           Begin           {начало блока 2}
             MaxF:= ValF;
             X1Max:= X1;     X2Max:= X2;    X3Max:= X3;
           end;            {конец блока 2}
         end;              {конец блока 1}
         X3:= X3 + 0.3;    {приращение переменной X3}
       end;                {конец цикла 3}
       X2:= X2 + 0.4;      {приращение переменной X2}
       end;                {конец цикла 2}
   end;                    {конец цикла 1}
```

```
   {вывод на экран результатов расчета}
   If MaxF>0 Then     {если возможен расчет значения функции}
   Begin
     Writeln('Максимальное значение функции F: ',MaxF:9:2);
     Writeln('при X1 = ',X1Max:3,' X2 = ',X2Max:4:1,' X3 = ',X3Max:4:1);
   end
   Else Writeln('При заданных ограничениях нет значений функции F.');
   Repeat Until KeyPressed;{ожидание нажатия любой клавиши}
End.                       {конец основной программы}
```

Результат работы программы

```
Введите значения ограничений Q1, Q2, Q3: 100 3 2.56
Максимальное значение функции F:     5651.80
при X1 =   1 X2 = 36.8  X3 = 29.9
```

Задача 2.24. Разработать схему алгоритма и программу для вычисления интервальной функции:

$$z=\begin{cases} e^{ax/3}+(\cos(x^3b)+2)^{1/3}, \text{ если } 1<x<10; \\ x^5+\cos^3(\ln(x)), \text{ если } 10\le x<10{,}8; \\ \sqrt{x^3}+\ln^3(xa), \text{ если } 10{,}8\le x<18{,}9; \\ \ln\left[\ln(x)\right], \text{если } x\ge 18{,}9, \end{cases}$$

где $a=\begin{cases} \sum\limits_{i=1}^{100}\dfrac{1}{i^2}, \text{если } x\ge 14, \\ \sum\limits_{i=1}^{33}\cos^3(i^2), \text{ если } x<14; \end{cases}$

$b=\sin^3(a)$.

Значение x вводится с клавиатуры. Значение функции z выводится на экран. При вычислении значения функции z необходимо учесть область определения функции. Если функция не определена, то вывести сообщение «Функция при заданном аргументе не определена».

Решение задачи

Схема алгоритма решения задачи представлена на рис. 2.24.

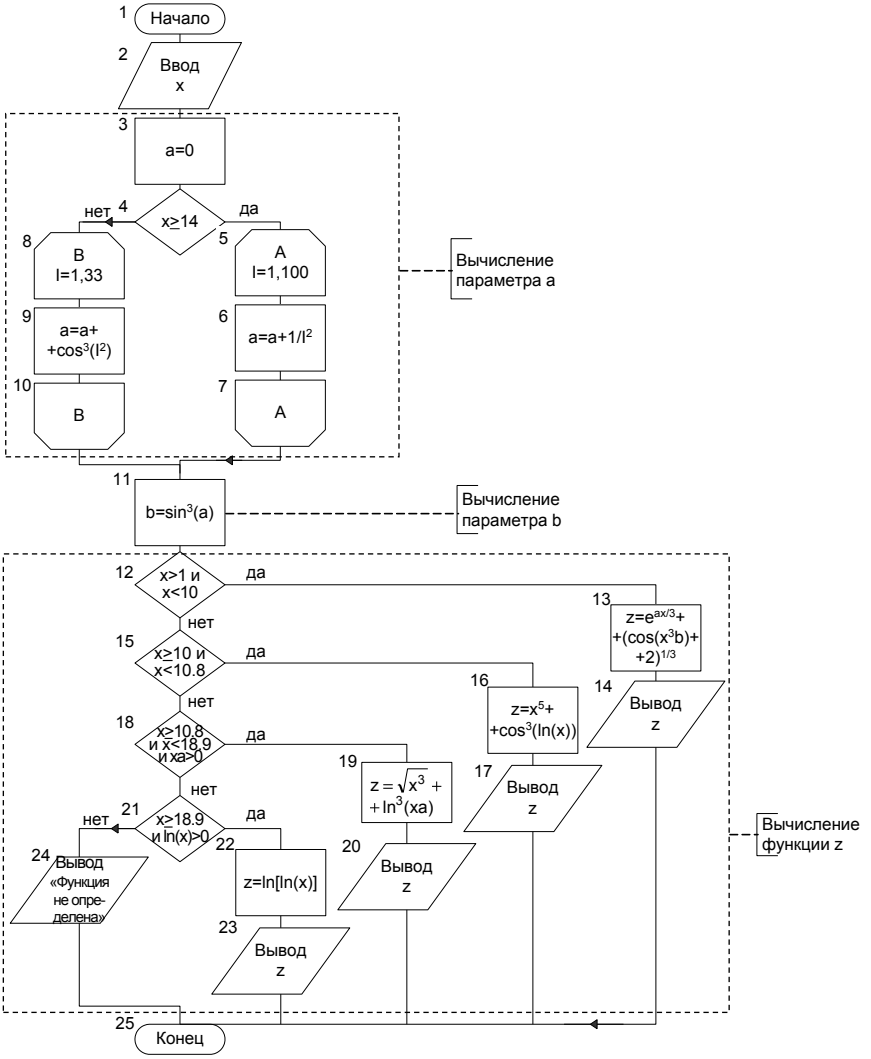


Рис. 2.24

Программа

```
Program Task2_24;
Uses Crt;
Var
  a, b, x, z : Real;
  I          : Integer;
Begin               {начало раздела операторов программы}
  ClrScr;           {очистка экрана}
  Write('Введите число: ');    Readln(x); {ввод x}
  {вычисление параметра a}
  a:= 0;
  If x >= 14 Then for I := 1 to 100 do a := a + 1/(I*I)
    Else for I := 1  to  33 do a:=a+Cos(I*I)*Cos(I*I)*Cos(I*I);
  b:= sin(a)*sin(a)*sin(a); {вычисление параметра b}
  {вычисление значения функции z}
  If (x>1) and (x<10) Then
  Begin                   {начало блока 1}
    z:= Exp(a*x/3) + Exp(1/3*Ln(Cos(x*x*x*b)+2));{вычисление z}
    Writeln('Значение функции равно ',z:10:3);{вывод z на экран}
  end                     {конец блока 1}
  Else If (x>=10) and (x<10.8) Then
       Begin              {начало блока 2}
       z:=Exp(5*Ln(x))+cos(ln(x))*cos(ln(x))*cos(ln(x));{вычисление z}
       Writeln('Значение функции равно ',z:10:3); {вывод z}
       end                {конец блока 2}
  Else If (x>=10.8) and (x<18.9) and (x*a>0) Then
        Begin                  {начало блока 3}
        {вычисление z}
        z:= Exp(1.5*Ln(x)) + Ln(x*a)*Ln(x*a)*Ln(x*a);
        Writeln('Значение функции равно ',z:10:3);{вывод z}
        end                {конец блока 3}
  Else If (x>=18.9) and (Ln(x)>0) Then
       Begin               {начало блока 4}
         z:= Ln(Ln(x));    {вычисление z}
         Writeln('Значение функции равно ',z:10:3);{вывод z}
       end                 {конец блока 4}
  Else Writeln('Функция при заданном аргументе не определена.');
  Readln;                  {ожидание нажатия клавиши Enter}
End.                       {конец программы}
```

Результат работы программы

```
Введите число: 1
Функция при заданном аргументе не определена.
Введите число: 100
```

```
Значение функции равно     1.527
Введите число: 9
Значение функции равно 13159.802
Введите число: 10.75
Значение функции равно 143562.559
Введите число: 15
Значение функции равно     90.853
```

3. ЧИСЛЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ НА ПЕРСОНАЛЬНОМ КОМПЬЮТЕРЕ

В разделе рассмотрены численные методы решения математических задач. Краткие теоретические сведения о численных методах, используемых при решении задач этого раздела, приведены в прил. 2.

3.1. Решение систем линейных уравнений

3.1.1. Задача с использованием матричного метода

Задача 3.1. Разработать схему алгоритма и программу решения системы линейных уравнений матричным методом:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = c_1; \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = c_2; \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = c_3. \end{cases}$$

Коэффициенты системы уравнений a_{11}, a_{12}, …, a_{33} формировать по закону $a_{ij} = \cos(i + j)$, свободные члены – по закону $c_i = \sin(i)$, i, $j = 1, 3$ (1). Результаты решения системы вывести на экран.

Решение задачи

Формульный аппарат матричного метода описан в прил. 2. В соответствии с постановкой задачи сформируем систему линейных уравнений:

$$\begin{cases} \cos(1+1)x_1 + \cos(1+2)x_2 + \cos(1+3)x_3 = \sin(1); \\ \cos(2+1)x_1 + \cos(2+2)x_2 + \cos(2+3)x_3 = \sin(2); \\ \cos(3+1)x_1 + \cos(3+2)x_2 + \cos(3+3)x_3 = \sin(3). \end{cases}$$

Схема алгоритма основной программы представлена на рис. 3.1*а*, схема алгоритма процедуры вычисления определителя союзной матрицы *DetSM* – на рис. 3.1*б*. Согласно этим схемам ниже представлен текст программы.

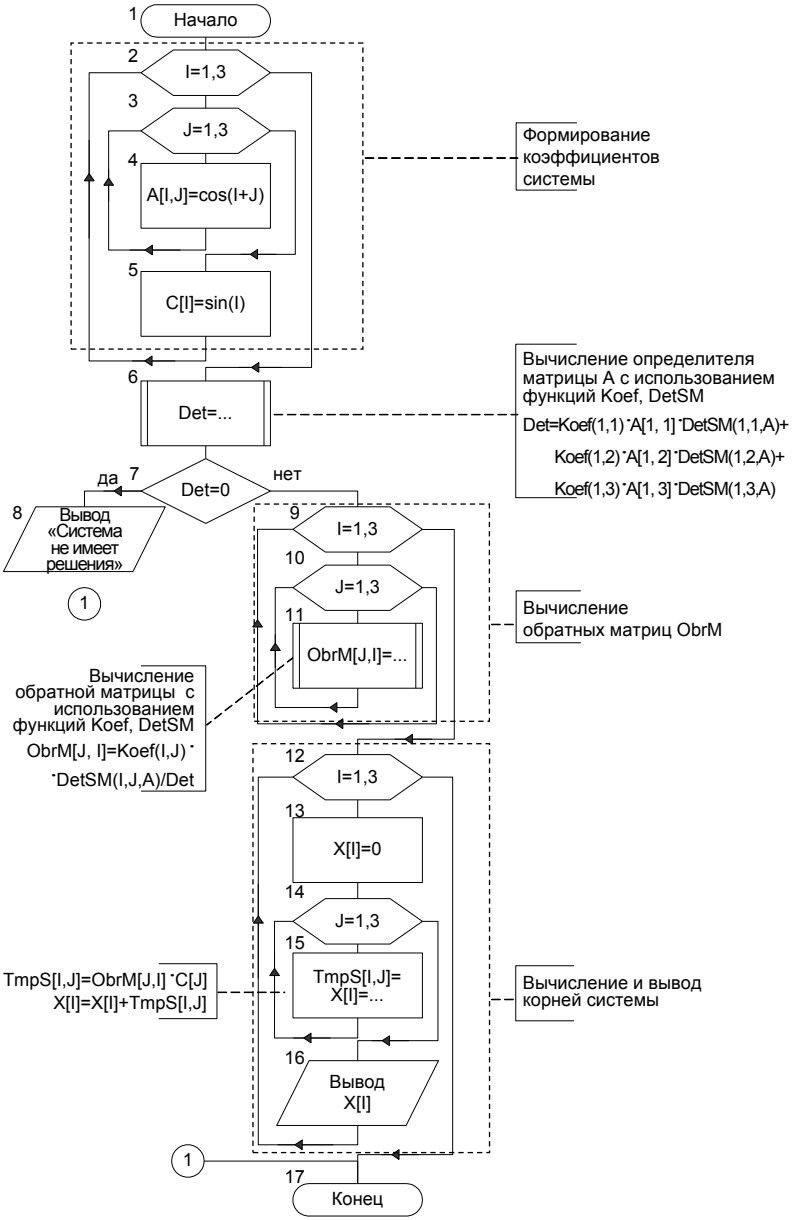

1
Начало
2
I=1,3
3
J=1,3
4
A[I,J]=cos(I+J)
5
C[I]=sin(I)
Формирование коэффициентов системы
6
Det=...
Вычисление определителя матрицы A с использованием функций Koef, DetSM
Det=Koef(1,1)*A[1, 1]*DetSM(1,1,A)+
Koef(1,2)*A[1, 2]*DetSM(1,2,A)+
Koef(1,3)*A[1, 3]*DetSM(1,3,A)
да
7
Det=0
нет
8
Вывод «Система не имеет решения»
1
9
I=1,3
10
J=1,3
11
ObrM[J,I]=...
Вычисление обратных матриц ObrM
Вычисление обратной матрицы с использованием функций Koef, DetSM
ObrM[J, I]=Koef(I,J) *
*DetSM(I,J,A)/Det
12
I=1,3
13
X[I]=0
14
J=1,3
15
TmpS[I,J]=ObrM[J,I] *C[J]
X[I]=X[I]+TmpS[I,J]
TmpS[I,J]= X[I]=...
Вычисление и вывод корней системы
16
Вывод X[I]
1
17
Конец

Рис. 3.1*а*

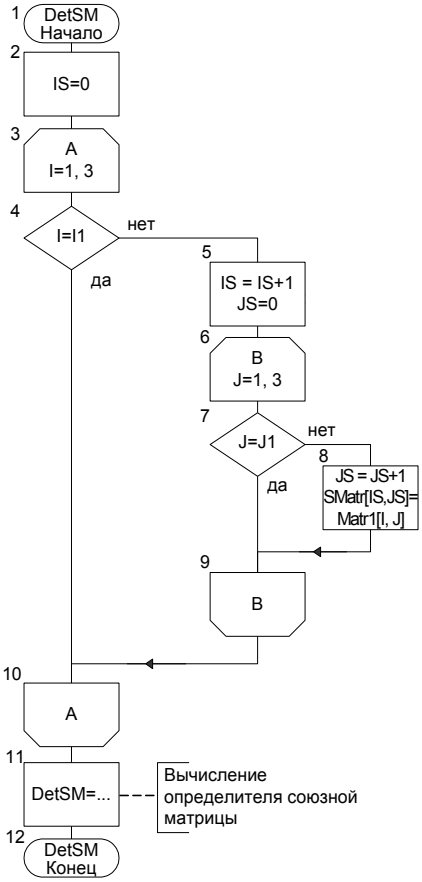


Рис. 3.1б

Пояснения к схеме алгоритма (см. рис. 3.1а)

Обозначения:

Det – определитель матрицы *A*;

ObrM – матрица, обратная для матрицы *A;*

X – массив корней системы уравнений;

TmpS – массив значений промежуточных слагаемых для вычисления корней системы;

DetSM – процедура вычисления определителя союзной матрицы.

Символ 1. Начало алгоритма.

Символы 2–5. Формирование коэффициентов и свободных членов системы.

Символ 6. Вычисление определителя матрицы *A*. В этом *символе* происходит вызов функций *Koef* и *DetSM*.

Символ 7. Проверка условия равенства определителя нулю. Если условие верно, то далее выполняются *символы 9–17*, если нет – *символ 8*.

Символ 8. Вывод на экран сообщения о том, что система не имеет решения.

Символы 9–11. Формирование обратной матрицы в двух циклах с параметрами *I*, *J*. В *символе 11* происходит вызов функций *Koef* и *DetSM*.

Символы 12–16. Вычисление корней системы *X*[*I*] в двух циклах с параметрами *I*, *J*. В *символе 13* производится первоначальное обнуление *X*[*I*]. В *символах 14–15* в цикле с параметром *J* производится вычисление *X*[*I*] суммированием *TmpS*[*I*, *J*]. *TmpS*[*I*, *J*] – это произведение элемента *ObrM*[*J*, *I*] обратной матрицы и значения свободного члена системы *C*[*J*]. В *символе 16* производится вывод значения корня *X*[*I*] на экран.

Символ 17. Конец алгоритма.

Пояснения к схеме алгоритма (см. рис. 3.1б)

Обозначения:

*Matr*1 – исходная матрица коэффициентов системы, переданная в процедуру *DetSM* в качестве формального параметра;

*I*1, *J*1 – номера строки и столбца исходной матрицы *Matr*1, для которых формируется союзная матрица;

SMatr – союзная матрица;

DetSM – определитель союзной матрицы;

I, *J* – параметры циклов, используемые для перебора строк и столбцов матрицы *Matr*1;

IS, *JS* – значения номеров строк и столбцов союзной матрицы *SMatr;*

Koef (I,J) – функция вычисления значения $(-1)^{I+J}$.

Символ 1. Начало алгоритма.

Символ 2. Обнуление номера строки союзной матрицы *SMatr* (*IS*).

Символ 3. Открытие цикла с параметром *I* = 1; 3 для перебора строк матрицы *Matr*1.

Символ 4. Проверка условия равенства номера строки матрицы *Matr*1 (*I*) номеру вычеркиваемой строки (*I*1). Если условие верно, то выполняется *символ 5*, если нет – *символ 10*.

Символ 5. Значение номера строки союзной матрицы *SMatr* (*IS*) увеличивается на 1. Переменная номера столбца союзной матрицы *SMatr* (*JS*) обнуляется.

Символ 6. Открытие цикла с параметром *J* = 1, 3 для перебора столбцов матрицы *Matr*1.

Символ 7. Проверка условия равенства номера столбца матрицы *Matr*1 (*J*) номеру вычеркиваемого столбца (*J*1). Если условие верно, то выполняется *символ 8*, если нет – *символ 9*.

Символ 8. Увеличение значения переменой *JS* на 1. В массив союзной матрицы *SMatr* заносится значение элемента матрицы *Matr*1[*I*, *J*].

Символ 9. Закрытие цикла с параметром *J*.

Символ 10. Закрытие цикла с параметром *I*.

Символ 11. Вычисление определителя союзной матрицы *SMatr*.

Символ 12. Конец алгоритма.

Программа

```
Program SLAU_Matr;
Uses Crt;
Type
   TMatr3=Array [1..3, 1..3] of Real; {тип матрицы коэффициентов}
Var
   I, J : Byte;
   Det, DetS : Real;
   A, TmpS, ObrM : TMatr3;
   C, X : Array [1..3] of Real;
{Функция определения значения коэффициента}
Function Koef(I1, J1 : Byte) : ShortInt;
begin        {начало раздела операторов функции Koef}
   {проверка на четность суммы (I1 + J1)}
   If ((I1 + J1) mod 2) = 0 Then Koef := 1 else Koef := -1;
end;         {конец функции Koef}

{Функция для вычисления определителя союзной матрицы
Входные данные:
Matr1 - исходная матрица,
I1, J1 - номера строки и столбца исходной матрицы Matr1,
для которых формируется союзная матрица
```

```
{Выходные данные: DetSM - определитель союзной матрицы}
Function DetSM(I1, J1 : Byte; Matr1 : TMatr3) : Real;
Var
   IS, JS, I, J : Byte;
   SMatr  : Array [1..2, 1..2] of Real; {союзная матрица}
begin               {начало раздела операторов функции DetSM}
   IS := 0;         {обнуление номера строки матрицы SMatr}
   for I:=1 to 3 do  {начало цикла 1-перебор строк матрицы Matr1}
     If I=I1 then  {вычеркиваем I1-тую строку матрицы Matr1}
     else begin           {начало блока 1}
          IS:=IS + 1; {увеличение номера строки матрицы SMatr}
          JS:=0;      {обнуление номера столбца матрицы SMatr}
       for J:=1 to 3 do    {начало цикла 2 - перебор столбцов Matr1}
          If J=J1 then   {вычеркиваем J1-ый столбец матрицы Matr1}
             else begin     {начало блока 2}
                    JS:=JS + 1;{увелич. номера столбца матрицы SMatr}
                    {формирование союзной матрицы}
                    SMatr[IS,JS]:=Matr1[I,J];
                  end;      {конец блока 2, конец цикла 2}
         end;       {конец блока 1, конец цикла 1}
    {определитель союзной матрицы}
    DetSM := SMatr[1,1]*SMatr[2,2] - SMatr[1,2]*SMatr[2,1];
end;           {конец функции DetSMatr}
Begin          {начало раздела операторов основной программы}
   ClrScr;               {очистка экрана}
   {формирование коэффициентов системы уравнений}
   for I:=1 to 3 do {начало цикла 1 - для перебора строк матрицы A}
   begin
     for J:=1 to 3 do {цикл 2 - для перебора столбцов матрицы A}
       A[I, J] := Cos(I+J);  {вычисление элемента A[I, J]}
     C[I] := Sin(I);
   end;                  {конец цикла 1}
   {вычисление определителя матрицы коэффициентов}
   Det := Koef(1, 1)*A[1,1] * DetSM(1, 1, A) +
          Koef(1, 2)*A[1, 2]* DetSM(1, 2, A) +
          Koef(1, 3)*A[1, 3]* DetSM(1, 3, A);
   If Det=0 then Writeln('Система не имеет решения.')
   else begin {начало блока 1- матрица коэф-тов не вырожденная}
          {формирование обратной матрицы}
          for I:=1 to 3 do
            for J:=1 to 3 do
              ObrM[J, I] := Koef(I, J) * DetSM(I, J, A)/Det;
          {вычисление корней системы}
          for I:=1 to 3 do {начало цикла 3-перебор корней X[I]}
          begin
            X[I] := 0;             {обнуление I-го корня}
            for J:=1 to 3 do       {начало цикла 4}
```

```
            begin
              TmpS[I,J]:=ObrM[I,J]*C[J]; {промежуточная сумма}
              X[I]:=X[I] + TmpS[I, J];   {вычисление I-го корня}
            end;                         {конец цикла 4}
            Writeln('X[', I, '] = ', X[I]:8:4); {вывод I-го корня}
          end;                 {конец цикла 3}
        end;                   {конец блока 1}
  Repeat Until KeyPressed;{ожидание нажатия любой клавиши}
End.                           {конец программы}
```

Результат работы программы

```
X[1] = -0.6875
X[2] = -0.9375
X[3] =  0.0625
```

3.1.2. Задача с использованием метода Крамера

***Задача** 3.2.* Разработать схему алгоритма и программу решения системы линейных уравнений с тремя неизвестными

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = c_1; \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = c_2; \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = c_3 \end{cases}$$

методом Крамера.

Коэффициенты и свободные члены вводить из файла. Результаты решения системы вывести на экран.

Решение задачи

Формульный аппарат метода Крамера описан в прил. 2. Схема алгоритма основной программы представлена на рис. 3.2. Схема алгоритма процедуры вычисления определителя союзной матрицы *DetSM* представлена на рис. 3.1*а*. Согласно этим схемам ниже представлен текст программы.

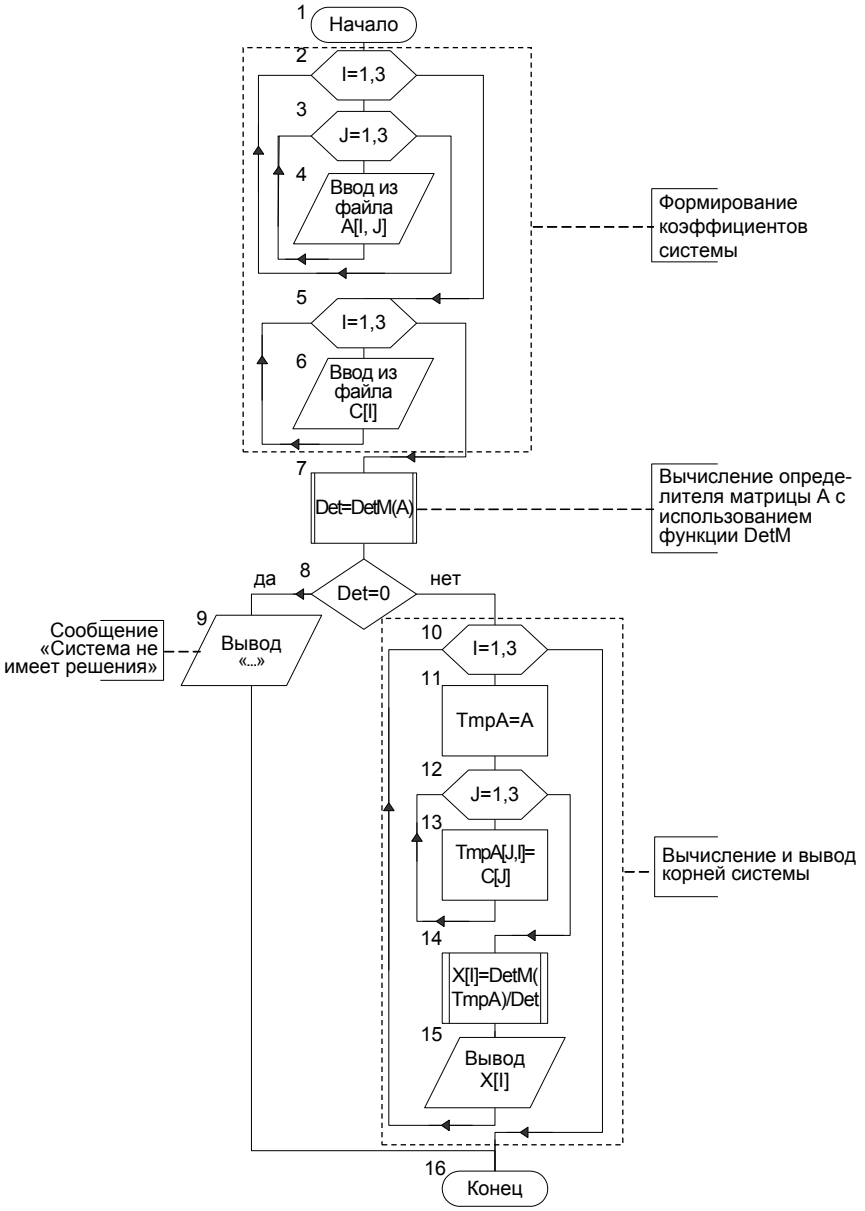

1
Начало
2
I=1,3
3
J=1,3
4
Ввод из файла A[I, J]
Формирование коэффициентов системы
5
I=1,3
6
Ввод из файла C[I]
7
Det=DetM(A)
Вычисление определителя матрицы A с использованием функции DetM
да
8
Det=0
нет
9
Вывод «...»
Сообщение «Система не имеет решения»
10
I=1,3
11
TmpA=A
12
J=1,3
13
TmpA[J,I]= C[J]
Вычисление и вывод корней системы
14
X[I]=DetM(TmpA)/Det
15
Вывод X[I]
16
Конец

Рис. 3.2

Пояснения к схеме алгоритма

Обозначения:

Det – определитель матрицы *A* (для вычисления вызывается функция *DetM*);

X – массив корней системы уравнений;

TmpA – промежуточная матрица, получаемая из исходной заменой столбца матрицы на столбец свободных членов;

Koef (I, J) – функция вычисления значения $(-1)^{I+J}$;

DetM(A), *DetM(TmpA)* – вызовы функции *DetM* для вычисления определителей матриц *A* и *TmpA*.

Символ 1. Начало алгоритма.

Символы 2–4. Ввод из файла коэффициентов *A*[*I*, *J*] системы в двух циклах с параметрами *I*, *J* .

Символы 5, 6. Ввод значений свободных членов *C*[*I*] системы в цикле с параметром *I*.

Символ 7. Вычисление определителя матрицы *A* (переменная *Det*). В этом *символе* происходит вызов функции *DetM*.

Символ 8. Проверка условия равенства определителя нулю. Если условие верно, то далее выполняется *символ 9*, если нет – *символы 10–16*.

Символ 9. Вывод на экран сообщения о том, что система не имеет решения.

Символы 10. Открытие внешнего цикла с параметром *I*.

Символ 11. Присвоение элементам матрицы *TmpA* значений элементов матрицы *A*.

Символы 12–13. Производится замена *I*-го столбца матрицы *TmpA* столбцом свободных членов *C*[*J*] в цикле с параметром *J*.

Символ 14. Вычисление корня *X*[*I*]. В этом *символе* происходит вызов функции *DetM*.

Символ 15. Вывод значения корня *X*[*I*] на экран.

Символ 16. Конец алгоритма.

Программа

```
Program SLAU_Kramer;
Uses Crt;
Type
```

```
   {тип матрицы коэффициентов системы уравнений}
   TMatr3 = Array [1..3, 1..3] of Real;
Var
   I, J : Byte;
   Det, TmpDet : Real;
   A, TmpA  : TMatr3;
   C, X : Array [1..3] of Real;
   F : Text;
{Функция определения значения коэффициента}
Function Koef(I1, J1 : Byte) : ShortInt;
Begin              {начало раздела операторов функции Koef}
   {проверка на четность суммы (I1 + J1)}
   If ((I1 + J1) mod 2) = 0 Then Koef := 1 else Koef := -1;
end;               {конец функции Koef}

{Функция для вычисления определителя союзной матрицы}
Function DetSM(I1, J1 : Byte; Matr1 : TMatr3) : Real;
begin              {начало раздела операторов функции DetSM}
   {текст функции DetSM приведен в разделе 3.1.1}
end;               {конец функции DetSM}

{Функция вычисления определителя матрицы
Входные данные: Matr1 - матрица
Выходные данные: DetMatr - определитель матрицы Matr1}
Function DetM(Matr1 : TMatr3) : Real;
Begin              {начало раздела операторов функции DetM}
   DetM := Koef(1, 1)*Matr1[1, 1]*DetSM(1, 1, Matr1) +
           Koef(1, 2)*Matr1[1, 2]*DetSM(1, 2, Matr1) +
           Koef(1, 3)*Matr1[1, 3]*DetSM(1, 3, Matr1);
end;               {конец функции DetM}

Begin        {начало раздела операторов основной программы}
   ClrScr;   {очистка экрана}
   {чтение из файла коэффициентов системы уравнений}
   Assign(F, 'koef.txt'); Reset(F); {открытие файла koef.txt}
   for I:=1 to 3 do
   begin     {начало цикла 1 - для перебора строк матрицы A}
     {чтение I-й строки матрицы A}
     for J:=1 to 3 do   Read(F, A[I, J]);
     Readln(F);            {переход на след. строку в файле}
   end;                    {конец цикла 1}
   {чтение из файла значений свободных членов}
   for I:= 1 to 3 do  Read(F, C[I]);
   Close(F);               {закрытие файла koef.txt}
   {расчет}
   Det := DetM(A);  {вычисление определителя матрицы A}
   If Det=0 then  Writeln('Система не имеет решения.')
```

```
   else begin {начало блока 1 - если матрица A не вырожденная}
          Writeln('Корни системы уравнений:');
          for I:=1 to 3 do
          begin       {начало цикла 2 - для перебора корней системы}
            {формирование коэф-тов I-й промежуточн. матрицы}
            TmpA := A;
            {замена I-го столбца свободными членами}
            for J:=1 to 3 do TmpA[J,I]:=C[J];
            X[I] := DetM(TmpA)/Det; {вычисление I-го корня}
            Writeln('X[', I, '] = ', X[I]:8:4);{вывод корня}
          end;                {конец цикла 6}
        end;                  {конец блока 1}
   Repeat Until KeyPressed; {ожидание нажатия любой клавиши}
End.                          {конец программы}
```

Результат работы программы

Файл *koef.txt*

```
7 4 3
6 4 5
4 -2 10
12 6 -1
```

```
Корни системы уравнений:
X[1] = 2.9811
X[2] = -1.0849
X[3] = -1.5094
```

3.1.3. Задача с использованием метода Гаусса

***Задача** 3.3.* Разработать схему алгоритма и программу решения системы линейных уравнений с тремя неизвестными

$$\begin{cases} 7x_1 + 4x_2 + 3x_3 = 12; \\ 6x_1 + 4x_2 + 5x_3 = 6; \\ 4x_1 - 2x_2 + 10x_3 = -1 \end{cases}$$

методом Гаусса.

Значения коэффициентов уравнений вводить с помощью клавиатуры. Результаты решения системы вывести на экран.

Решение задачи

Формульный аппарат метода Гаусса описан в прил. 2. Схема алгоритма процедуры *Gauss* представлена на рис. 3.3. Согласно этой схеме представлен текст программы.

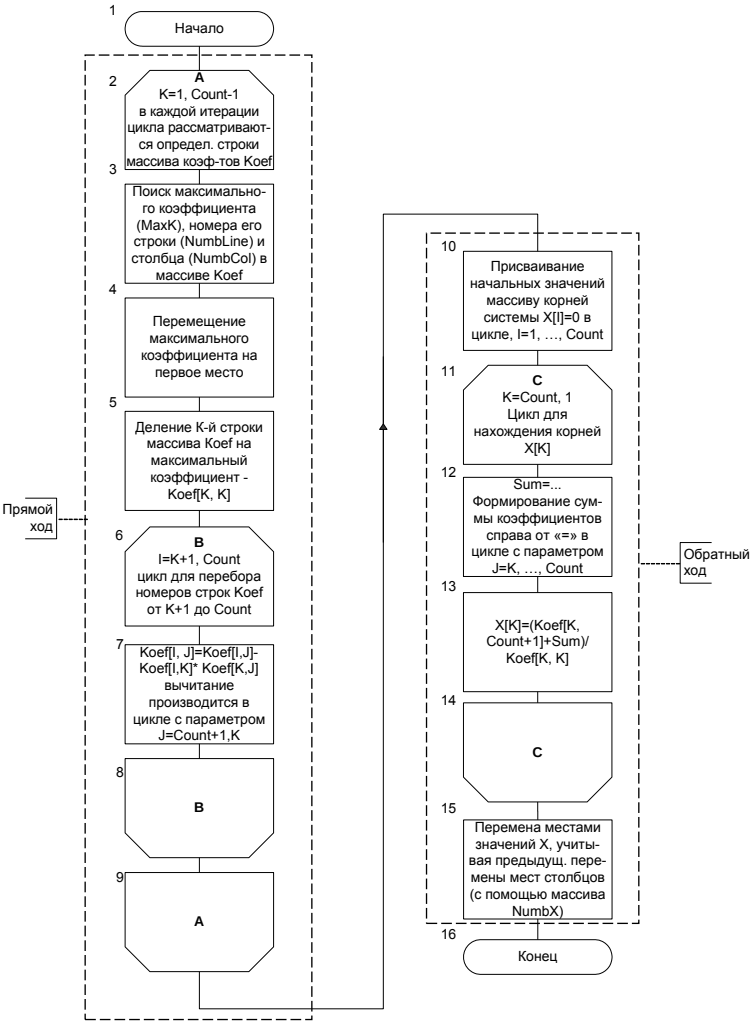


Рис. 3.3

Обозначения схемы алгоритма

Koef – массив коэффициентов системы уравнений;

Count – число строк массива *Koef* (число уравнений системы);

X[*I*] – массив значений корней системы уравнений, *I* = 1, …, *Count*;

NumbX[*I*] – массив порядковых номеров значений переменных массива *X*, а именно, *NumbX*[1] – содержит индекс переменной *X*[1], *NumbX*[2] – индекс *X*[2]. Например, *NumbX*[1] = 3 означает, что значение переменной *X*[*NumbX*[1]] соответствует x_3.

Программа

```
Program SLAU_Gauss;
Uses Crt;
Type
   Tablica = Array [1..3, 1..4] of Real;
   Colonka = Array [1..3] of Real;
Var
   Count : Integer;
   A : Tablica;
   X : Colonka;
   flag : Boolean;
   I, J : Integer;
{Функция вычисления корней системы уравнений методом Гаусса
Входные данные: Koef – матрица коэффициентов системы,
                Count – число уравнений системы.
Выходные данные: X – массив значений корней системы
                flag_virogdena – флаг вырожденной матрицы}
Procedure Gauss(Koef : Tablica; Count : Byte;
           Var X: Colonka; Var flag_virogdena : Boolean);
Var
   I, J, K, NumbLine, NumbCol, TmpNmb : Integer;
   Sum, MaxK : Real;
   Temp : Array [1..4] of Real;
   NumbX : Array[1..3] of integer;
Begin  {начало раздела операторов процедуры Gauss}
   flag_virogdena:=false;  {начальное значения флага}
   {начальное сохранение порядка следования переменных X}
   for I:=1 to Count do NumbX[I]:= I;
   {прямой ход решения}
   for K:=1 to Count-1 do
   Begin      {начало цикла 1 – для перебора массивов Koef}
     {поиск максимума массива Koef}
     MaxK:= Koef[K, K]; {первоначальное значение максимума}
```

```
  NumbLine:= K; {первонач. присвоение номеру строки максимума}
  NumbCol:= K; {первонач. присвоение номеру столбца максимума}
  for I:=K to Count do
    for J:=K to Count do
      If Abs(Koef[I, J]) > Abs(MaxK) Then
      Begin       {начало блока 1}
        MaxK:= Koef[I, J]; NumbLine:= I; NumbCol:= J;
      end;        {конец блока 1}
  If MaxK=0 Then flag_virogdena:=true; {матрица вырожденная}
  If not(flag_virogdena) Then {если матрица невырожденная}
  Begin           {начало блока 2}
  {перемена местами K-й строки и строки NumbLine}
    for J:=K to Count+1 do
    Begin {начало цикла 2-для перебора значений по столбцам}
      Temp[J]:= Koef[NumbLine, J];
      Koef[NumbLine, J]:= Koef[K, J];
      Koef[K, J]:= Temp[J];
    end;          {конец цикла 2}
    {перемена местами K-го столбца и столбца NumbCol}
    for I:=1 to Count do
    Begin {начало цикла 3-для перебора значений по строкам}
      Temp[I]:= Koef[I, NumbCol];
      Koef[I, NumbCol]:= Koef[I, K];
      Koef[I, K]:= Temp[I];
    end;          {конец цикла 3}
    {уточнение порядка следования переменных}
    TmpNmb:= NumbX[NumbCol]; NumbX[NumbCol]:= NumbX[K];
    NumbX[K]:= TmpNmb;
    {деление K-й строки на 1-й коэф-т строки Koef[K,1]}
    for I:=Count+1 downto K do
      Koef[K, I]:=Koef[K, I]/Koef[K, K];
    {исключение переменной K+1-го уравнения}
    for I:=K+1 to Count do
      for J:=Count+1 downto K do
        Koef[I, J]:= Koef[I, J]-Koef[I, K]*Koef[K, J];
  end;            {конец блока 2}
end;              {конец цикла 1 - прямого хода}
{обратный ход}
for I:=1 to Count do X[I]:= 0; {начальные значения X[I]}
If Koef[Count, Count]=0 Then flag_virogdena:= True;
for K:=Count downto 1 do
Begin {начало цикла 4 - для перебора строк массива Koef}
  Sum:=0;         {первоначальное обнуление суммы}
  {вычисление суммы коэффициентов справа от "="}
  for J:=K to Count do Sum:= Sum - Koef[K, J]*X[J];
  {вычисление значения переменной X}
  If Koef[K, K] <> 0 Then
```

```
                    X[K]:=(Koef[K, Count+1] + Sum)/Koef[K, K]
                        Else X[K]:= 0;
    end;        {конец цикла 4}
    {перемена местами X, учитывая предыдущие перемены столбцов}
    for I:=1 to Count do Temp[I]:= X[I];
    for I:=1 to Count do X[NumbX[I]]:= Temp[I];
end;           {конец процедуры Gauss}

Begin  {начало раздела операторов основной программы}
   ClrScr;
   for I:=1 to 3 do
   Begin {начало цикла 1 – для перебора строк матрицы A}
     {ввод коэффициентов I-го уравнения}
     Write('Введите коэффициенты ', I,'-го уравнения: ');
     for J:=1 to 4 do Read(a[I, J]);
     Readln;          {переход на след. строку}
   end;               {конец цикла 1}
   Gauss(a, 3, X, flag);{вызов процедуры Gauss для вычисления решения}
   {вывод результатов}
   If not flag then for I:=1 to 3 do
                    Writeln('x',I,'=', x[I]:7:4)
               Else Writeln('Система не имеет решения.')
   Readln;          {ожидание нажатия клавиши Enter}
End.                {конец основной программы}
```

Результат работы программы

```
Введите коэффициенты 1-го уравнения: 7 4 3 12
Введите коэффициенты 2-го уравнения: 6 4 5 6
Введите коэффициенты 3-го уравнения: 4 -2 10 -1
x1 =  2.9811
x2 = -1.0849
x3 = -1.5094
```

3.2. **Решение нелинейных уравнений**

Краткие теоретические сведения о методах решения нелинейных уравнений (методы дихотомии, хорд и касательных) изложены в прил. 2.

3.2.1. Задача с использованием метода дихотомии

***Задача** 3.4.* Разработать схему алгоритма и программу решения нелинейного уравнения $f(x) = e^x - 10\sin x$ на отрезке [–10, 4] мето-

дом дихотомии с погрешностью не более 0,005. В процессе решения построить график, выбрать отрезок для уточнения корня.

Решение задачи

Определим интервалы, на которых функция $f(x)$ меняет знак, для чего проведем табуляцию функции на отрезке [–10, 4] с произвольным шагом.

x	–10	–9	–8	–7	–6	–5	–4	–3
$f(x)$	–5,44	4,12	9,89	6,57	–2,79	–9,58	–7,54	1,46
x	–2	–1	0	1	2	3	4	
$f(x)$	9,23	8,78	1	–5,69	–1,7	18,67	62,16	

Из таблицы видно, что на отрезках [–10, –9], [–7, –6], [–4, –3], [0, 1], [2, 3] меняется знак функции, значит, на этих отрезках есть корни уравнения. Определим корень уравнения на отрезке [0, 1]. На рис. 3.4, *а* представлен график функции нелинейного уравнения $y = e^x - 10\sin x$ на отрезке [–10, 4]. Схема алгоритма решения задачи представлена на рис. 3.4, *б*.

Программа

```
Program Dihotom;
Uses Crt;
Var
  Xn, Xk, e, X : Real;
{функция вычисления значения нелинейного уравнения}
Function F(X : Real) : Real;
Begin                {начало раздела операторов функции F}
  F:= Exp(X) - 10*Sin(X);
end;                 {конец функции F}
Begin                {начало раздела операторов программы}
  ClrScr;            {очистка экрана}
  Xn:= 0; Xk:= 1;    {задание границ отрезка}
  e:= 0.005;         {значение погрешности вычисления}
  While Abs(Xk - Xn)/2 > e do
  Begin              {начало цикла}
    X:= (Xn+Xk)/2;   {новое значение границы отрезка}
    {проверка меняет ли функция знак на отрезке [Xn, X]}
    If F(X)*F(Xk) <= 0 Then Xn:= X {отрезок для анализа [X,Xk]}
                       Else Xk:= X;{отрезок для анализа [Xn,X]}
```

```
    Writeln('X =', X:8:5, ' F(X) = ', F(X):5:3, ' Xn =',
Xn:8:5, ' Xk =', Xk:8:5);     {вывод результатов}
  end;                {конец цикла}
  Repeat Until KeyPressed; {ожидание нажатия любой клавиши}
end.                  {конец программы}
```

Результат работы программы

```
X = 0.50000 F(X) = -3.146 Xn = 0.00000 Xk = 0.50000
X = 0.25000 F(X) = -1.190 Xn = 0.00000 Xk = 0.25000
X = 0.12500 F(X) = -0.114 Xn = 0.00000 Xk = 0.12500
X = 0.06250 F(X) =  0.440 Xn = 0.06250 Xk = 0.12500
X = 0.09375 F(X) =  0.162 Xn = 0.09375 Xk = 0.12500
X = 0.10938 F(X) =  0.024 Xn = 0.10938 Xk = 0.12500
X = 0.11719 F(X) = -0.045 Xn = 0.10938 Xk = 0.11719
```

Пояснения к схеме алгоритма

Обозначения:

Xn, *Xk* – значения левой и правой границ отрезка, на котором производится поиск корней нелинейного уравнения;

e – значение погрешности;

F(X), *F(Xk)* – функция, которая вычисляет значение нелинейного уравнения $f(x) = e^x - 10\sin x$ в точках *X*, *Xk*.

Символ 1. Начало алгоритма.

Символ 2. Присвоение переменным границ отрезка (*Xn*, *Xk*) и переменной погрешности вычислений (*e*) начальных значений.

Символ 3. Проверка условия окончания вычислений: величина отрезка, на котором производится поиск корня, больше величины $2 \cdot e$. Если условие верно, выполняется *символ 4*, иначе – *символ 9.*

Символ 4. Вычисление нового приближения к корню уравнения (*X*).

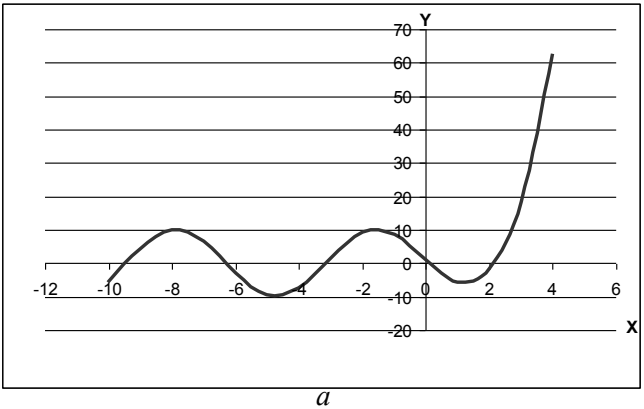
Символ 5. Проверка знаков функции на концах отрезка [*X*, *Xk*]. Если выражение $F(X) \cdot F(Xk) < 0$, то на этом отрезке функция меняет знак и имеется корень; в этом случае выполняется *символ 6*. Если выражение $F(X) \cdot F(Xk) \geq 0$, то функция на концах отрезка имеет одинаковый знак и, значит, корня на этом отрезке нет; в этом случае выполняется *символ 7*.

Символ 6. Присвоение переменной начала отрезка (*Xn*) значения *X*.

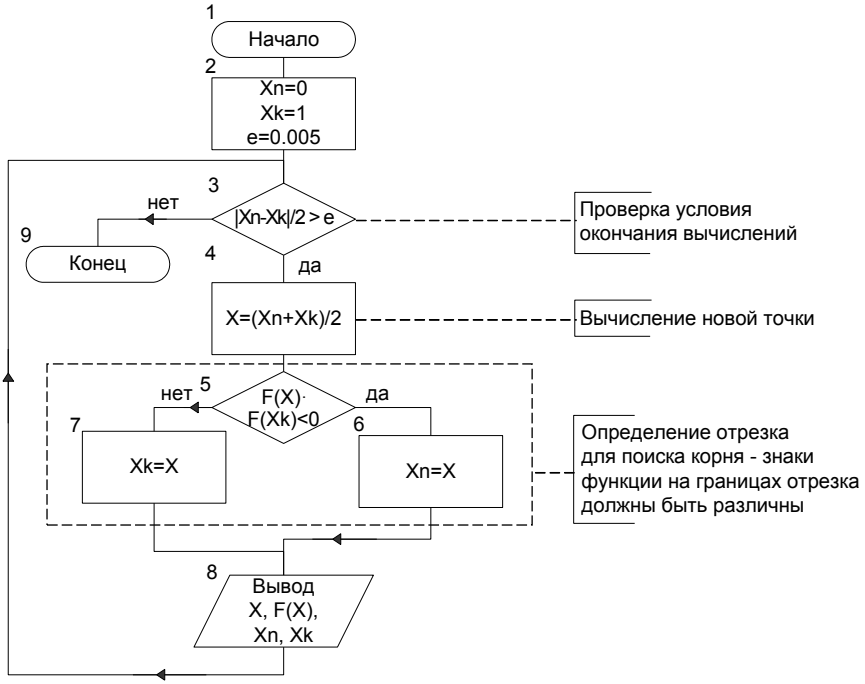
Символ 7. Присвоение переменной конца отрезка (*Xk*) значения *X*.

Символ 8. Вывод результатов на экран.

Символ 9. Конец алгоритма.


Y
70
60
50
40
30
20
10
0
-10
-20
-12
-10
-8
-6
-4
-2
0
2
4
6
X

а


1
Начало
2
Xn=0
Xk=1
e=0.005
3
нет
|Xn-Xk|/2 > e
Проверка условия окончания вычислений
9
Конец
4
да
X=(Xn+Xk)/2
Вычисление новой точки
нет
5
F(X)·
F(Xk)<0
да
7
Xk=X
6
Xn=X
Определение отрезка для поиска корня - знаки функции на границах отрезка должны быть различны
8
Вывод
X, F(X),
Xn, Xk

б

Рис. 3.4

3.2.2. Задача с использованием метода хорд

***Задача** 3.5.* Разработать схему алгоритма и программу решения нелинейного уравнения $f(x) = e^x - 10\sin x$ на отрезке [0, 1] методом хорд с погрешностью не более 0,005.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 3.5.

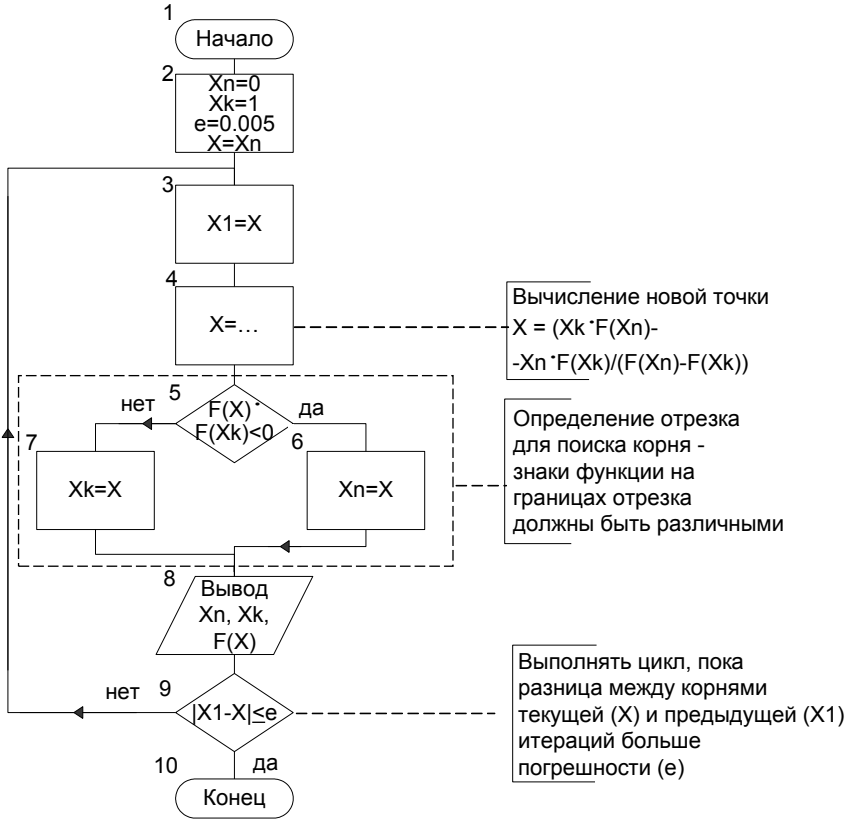


Рис. 3.5

Пояснения к схеме алгоритма

Обозначения:

Xn, *Xk* – значения левой и правой границ отрезка, на котором производится поиск корней нелинейного уравнения;

F(Xn), *F(Xk)*, *F(X)* – значения функции нелинейного уравнения $f(x) = e^x - 10\sin x$ соответственно в точках *Xn*, *Xk*, *X*.

Символ 1. Начало алгоритма.

Символ 2. Присвоение начальных значений переменным границы отрезка (*Xn*, *Xk*), переменной *X* и переменной погрешности вычислений (*e*).

Символ 3. Присвоение переменной *X*1 предыдущего приближения к корню уравнения.

Символ 4. Вычисление нового приближения к корню уравнения (*X*).

Символ 5. Проверка знаков функции на концах отрезка [*X*, *Xk*]. Если выражение $F(X)\cdot F(Xk) < 0$, то на этом отрезке функция меняет знак и имеется корень; в этом случае выполняется *символ 6*. Если выражение $F(X)\cdot F(Xk) \geq 0$, то функция на концах отрезка имеет одинаковый знак и, значит, корня на этом отрезке нет; в этом случае выполняется *символ 7*.

Символ 6. Присвоение переменной начала отрезка (*Xn*) значения *X*.

Символ 7. Присвоение переменной конца отрезка (*Xk*) значения *X*.

Символ 8. Вывод результатов на экран.

Символ 9. Проверка условия окончания вычислений: разница между текущим и предыдущим приближениями к корню уравнения меньше погрешности (*e*). Если условие верно, то выполняется *символ 10*, иначе – *символ 3*.

Символ 10. Конец алгоритма.

Программа

```
Program Chord;
Var
   Xn, Xk, X, X1, e : Real;
{вычисление значения функции}
Function F(X : Real) : Real;
Begin        {начало раздела операторов функции F}
   F := Exp(X) - 10*sin(X);
end;         {конец функции F}
```

```
Begin          {начало раздела операторов основной программы}
   Xn := 0; Xk := 1; {задание границ отрезка}
   e := 0.0005;      {значение погрешности вычислений}
   X1 := Xn;
   Repeat            {начало цикла поиска корня}
     X1 := X;        {сохранение корня предыдущей итерации }
     {новое значение границы отрезка}
     X := (Xk*F(Xn) - Xn*F(Xk))/(F(Xn) - F(Xk));
     {проверка, меняет ли функция знак на отрезке [Xn, X]}
     If F(X)*F(Xk) < 0 Then Xn := X {отрезок для анализа [X,Xk]}
                       Else Xk := X;{отрезок для анализа [Xn,X]}
     {вывод результатов}
     Writeln(' X = ', X :10:8, ' Значение функции = ', F(X):10:8);
   Until Abs(X1 - X) <= e;        {конец цикла}
End.                              {конец программы}
```

Результат работы программы

```
X = 0.14933335 Значение функции = -0.32672935
X = 0.11255751 Значение функции = -0.00406326
X = 0.11210201 Значение функции = -0.00004662
```

3.2.3. Задача с использованием метода касательных

***Задача** 3.6.* Разработать схему алгоритма и программу решения нелинейного уравнения $f(x) = e^x - 10\sin x$ на отрезке [0, 1] методом касательных с погрешностью не более 0,005.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 3.6.

Пояснения к схеме алгоритма

Обозначения:

Xn, *Xk* – значения левой и правой границ отрезка, на котором производится поиск корней нелинейного уравнения;

e – значение погрешности;

F(Xk), *F(Xn)*, *F(X)* – значения функции нелинейного уравнения $f(x) = e^x - 10\sin x$ соответственно в точках *Xk*, *Xn*, *X*;

FP1(Xn), *FP1(Xk)* – значения первой производной функции нелинейного уравнения $f(x) = e^x - 10\sin x$ соответственно в точках *Xn*, *Xk*;

FP2(Xn) – значение второй производной функции нелинейного уравнения $f(x) = e^x - 10\sin x$ в точке *Xn*;

X1 – значение корня на предыдущей итерации цикла.

Символ 1. Начало алгоритма.

Символ 2. Присвоение переменным границы отрезка (*Xn*, *Xk*), переменной *X* и переменной погрешности вычислений (*e*) начальных значений.

Символ 3. Присвоение переменной *X*1 предыдущего приближения к корню уравнения.

Символ 4. Проверка знаков функции и второй производной в точке *Xn*. Если выражение $F(Xn)\cdot FP2(Xn) > 0$, следовательно, функция и вторая производная в точке *Xn* имеют одинаковые знаки и в этом случае выполняется *символ 5*. Если выражение $F(Xn)\cdot FP2(Xn) \le 0$, следовательно, функция и вторая производная в точке *Xn* имеют разные знаки и в этом случае выполняется *символ 6*.

Символ 5. Вычисление нового приближения к корню уравнения (*X*) путем построения касательной в точке *Xn*.

Символ 6. Вычисление нового приближения к корню уравнения (*X*) путем построения касательной в точке *Xk*.

Символ 7. Проверка знаков функции на концах отрезка [*X*, *Xn*]. Если выражение $F(X)\cdot F(Xk) < 0$, то на этом отрезке функция меняет знак и имеется корень; в этом случае выполняется *символ 8*. Если выражение $F(X)\cdot F(Xk) \ge 0$, то функция на концах отрезка имеет одинаковый знак и, значит, корня на этом отрезке нет; в этом случае выполняется *символ 9*.

Символ 8. Присвоение переменной начала отрезка (*Xn*) значения *X*.

Символ 9. Присвоение переменной конца отрезка (*Xk*) значения *X*.

Символ 10. Вывод результатов на экран.

Символ 11. Проверка условия окончания вычислений: разница между текущим и предыдущим приближениями к корню уравнения меньше погрешности (*e*). Если условие верно, то выполняется *символ 12*, иначе – *символ 3*.

Символ 12. Конец алгоритма.

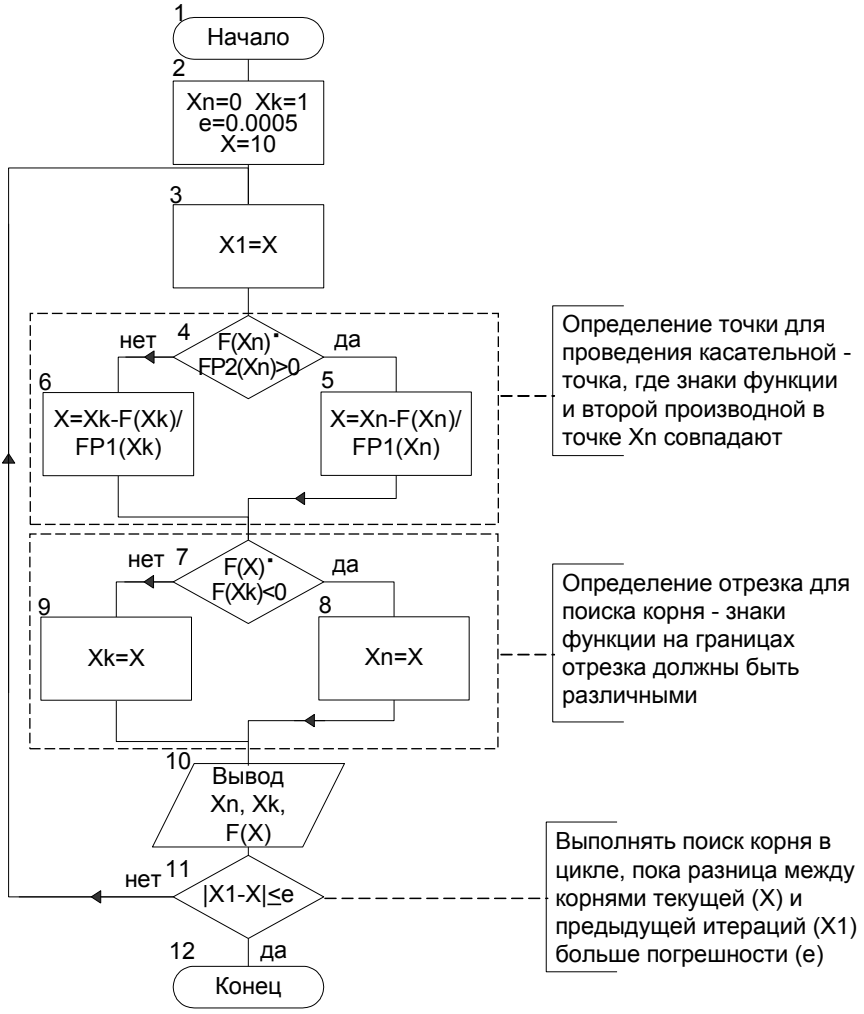


Рис. 3.6

Программа

```
Program Kasat;
Uses Crt;
Var
   Xn, Xk, X, X1, e : Real;
```

```
{вычисление значения функции}
Function F(X : Real) : Real;
Begin                {начало раздела операторов функции F}
   F := Exp(X) - 10*sin(X);
end;                 {конец функции F}
{вычисление значения первой производной функции}
Function FP1(X : Real) : Real;
Begin                {начало раздела операторов функции FP}
   FP1 := Exp(X) - 10*cos(X);
end;                 {конец функции FP1}
{вычисление значения второй производной функции}
Function FP2(X : Real) : Real;
Begin                {начало раздела операторов функции FP2}
   FP2 := Exp(X) + 10*sin(X);
end;                 {конец функции FP2}
Begin                {начало раздела операторов основной программы}
   ClrScr;           {функция очистки экрана}
   Xn := 0; Xk := 1;{задание границ отрезка}
   e := 0.0005;      {значение погрешности вычислений}
   X := 10;
   Repeat            {начало цикла}
     X1 := X;        {сохранение корня предыдущей итерации}
     {определение точки для проведения касательной -
     знаки функции и второй производной должны совпадать}
     If F(Xn)*FP2(Xn) > 0 Then X := Xn - F(Xn)/FP1(Xn)
                          Else X := Xk - F(Xk)/FP1(Xk);
     {определение отрезка-знаки функции на концах отрезках различны}
     If F(X)*F(Xk)<0 Then Xn:=X {отрезок для анализа [X,Xk]}
                     Else Xk:=X;{отрезок для анализа [Xn,X]}
     Writeln(' Xn = ', Xn :8:8, ' Xk = ', Xk :5:3, ' Значение
функции = ', F(X):13:11);       {вывод результатов на экран}
     {конец цикла-разница между корнями < погрешности}
   Until Abs(X-X1) < e;
End.                             { конец программы}
```

Результат работы программы

```
Xn = 0.11111111 Xk = 1.000 Значение функции = 0.00869278363
Xn = 0.11209660 Xk = 1.000 Значение функции = 0.00000108286
Xn = 0.11209672 Xk = 1.000 Значение функции = 0.00000000000
```

3.3. **Аппроксимация табличных данных методом наименьших квадратов**

***Задача** 3.7.* Разработать схему алгоритма и программу аппроксимации функции, заданной таблично, методом наименьших квадратов с помощью полинома второй степени $F(x) = a_0 + a_1 x + a_2 x^2$.

x	0	1	2	3	4	5	6	7	8	10
$F(x)$	–1	1	5	17	31	49	71	97	112	144

Решение задачи

Формульный аппарат аппроксимации методом наименьших квадратов описан в прил. 2.

$$t_0 = 10;$$

$$t_1 = \sum_{i=1}^{10} x_i = 0+1+2+3+4+5+6+7+8+10 = 46;$$

$$t_2 = \sum_{i=1}^{10} x_i^2 = 0^2+1^2+2^2+3^2+4^2+5^2+6^2+7^2+8^2+10^2 = 304;$$

$$t_3 = \sum_{i=1}^{10} x_i^3 = 0^3+1^3+2^3+3^3+4^3+5^3+6^3+7^3+8^3+10^3 = 2296;$$

$$t_4 = \sum_{i=1}^{10} x_i^4 = 0^4+1^4+2^4+3^4+4^4+5^4+6^4+7^4+8^4+10^4 = 18772;$$

$$c_0 = \sum_{i=1}^{10} y_i = -1+1+5+17+31+49+71+97+112+144 = 526;$$

$$c_1 = \sum_{i=1}^{10} x_i y_i = 0\cdot(-1)+1\cdot 1+2\cdot 5+3\cdot 17+4\cdot 31+5\cdot 49+6\cdot 71+7\cdot 97+$$
$$+8\cdot 112+10\cdot 144 = 3872;$$

$$c_2 = \sum_{i=1}^{10} x_i^2 y_i = 0^2 \cdot (-1) + 1^2 \cdot 1 + 2^2 \cdot 5 + 3^2 \cdot 17 + 4^2 \cdot 31 + 5^2 \cdot 49 + 6^2 \cdot 71 +$$

$$+ 7^2 \cdot 97 + 8^2 \cdot 112 + 10^2 \cdot 144 = 30772.$$

Таким образом, линейная система имеет вид

$$\begin{cases} 10a_0 + 46a_1 + 304a_2 = 526; \\ 46a_0 + 304a_1 + 2296a_2 = 3872; \\ 304a_0 + 2296a_1 + 18772a_2 = 30772. \end{cases}$$

Для решения полученной линейной системы используется метод Гаусса (п. 3.1.3). Схема алгоритма решения задачи представлена на рис. 3.7.

Программа

```
Program Approksim;
Type
   Tablica = Array [1..3, 1..4] of Real;
   Colonka = Array [1..3] of Real;
Var
   I : Byte;
   T0, T1, T2, T3, T4, C0, C1, C2 : Integer;
   X, Y : Array[1..10] of Integer;
   A : Tablica;
   XK : Colonka;
{функция вычисления корней системы уравнений методом Гаусса}
Procedure Gauss(Koef : Tablica; Count : Byte;
          Var X: Colonka; Var flag_virogdena : Boolean);
Var
   …
Begin  {начало раздела операторов процедуры Gauss}
   {текст программы функции представлен в разделе 3.1.3}
End;          {конец функции Gauss}
Begin         {начало раздела операторов основной программы}
   T0 := 10;
   For I := 1 to 10 do              {цикл по всем точкам}
   Begin                            {начало цикла}
     Write('Введите значения X, Y для точки ', I, ': ');
     Readln(X[I], Y[I]);   {чтение значений X[I], Y[I]}
     {вычисление коэффициентов T1, T2}
```

```
      T1 := T1+X[I]; T2 := T2 + X[I]*X[I];
      T3 := T3 + X[I]*X[I]*X[I];     {вычисление коэффициента T3}
      T4 := T4 + X[I]*X[I]*X[I]*X[I]; {вычисление коэффициента T4}
      {вычисление коэффициентов C0, C1, C2}
      C0:=C0 + Y[I];     C1:=C1 + X[I]*Y[I];
      C2:=C2 + X[I]*X[I]*Y[I];
   end;                              {конец цикла}
   {коэффициенты 1-го уравнения системы}
   A[1, 1]:=T0;  A[1, 2]:=T1;  A[1, 3]:=T2;  A[1, 4]:=C0;
   {коэффициенты 2-го уравнения системы}
   A[2, 1]:=T1;  A[2, 2]:=T2;  A[2, 3]:=T3;  A[2, 4]:=C1;
   {коэффициенты 3-го уравнения системы}
   A[3, 1]:=T2;  A[3, 2]:=T3;  A[3, 3]:=T4;  A[3, 4]:=C2;
   Gauss(A, 3, XK, B);{вычисление корней системы уравнений}
   {вывод значений корней}
   If not B Then for I :=1 to 3 do
                    Writeln('x',I, ' = ', XK[I]:5:2)
            Else Writeln('Система не имеет решения.');
End.                 {конец основной программы}
```

Результат работы программы

```
Введите значения X, Y для точки 1: 0 -1
Введите значения X, Y для точки 2: 1 1
Введите значения X, Y для точки 3: 2 5
Введите значения X, Y для точки 4: 3 17
Введите значения X, Y для точки 5: 4 31
Введите значения X, Y для точки 6: 5 49
Введите значения X, Y для точки 7: 6 71
Введите значения X, Y для точки 8: 7 97
Введите значения X, Y для точки 9: 8 112
Введите значения X, Y для точки 10: 10 144
x1 = -7.86
x2 =  7.66
x3 =  0.83
```

Пояснения к схеме алгоритма

Обозначения:

X[*I*], *Y*[*I*] – массивы значений *x* и *F*(*x*) (заданные таблично);

Gauss(A, 3, XK, B) – вызов процедуры *Gauss* для вычисления корней системы уравнений с тремя переменными методом Гаусса; входными параметрами процедуры являются: матрица коэффициентов системы *A*, число уравнений системы 3, выходными параметра-

ми: массив значений корней системы XK, флаг существования решения системы – переменная *B* (равна false, если матрица коэффициентов невырожденная и система имеет решение).

Символ 1. Начало алгоритма.

Символы 2–3. Присвоение первоначальных значений коэффициентам *T*0, *T*1, *T*2, *T*3, *T*4, *C*0, *C*1, *C*2.

Символы 4–8. В цикле с параметром *I* = 1; 10 производится ввод значений элементов массивов *X*[*I*] и *Y*[*I*] и вычисление коэффициентов *T*0, *T*1, *T*2, *T*3, *T*4, *C*0, *C*1, *C*2 с использованием суммирования.

Символы 9–11. Формирование массива коэффициентов *A*, содержащего коэффициенты системы уравнений (передаются в качестве входного параметра в процедуру *Gauss*).

Символ 12. Вызов процедуры *Gauss*, реализующей решение системы линейных уравнений методом Гаусса.

Символ 13. Проверка условия существования решения системы уравнений (*B = false*).

Символ 14. Вывод на экран сообщения «Система не имеет решения».

Символ 15. Вывод значений корней системы уравнений производится в цикле (табл. П 1.13), на схеме для упрощения вывод представлен в одном символе.

Символ 16. Конец алгоритма.

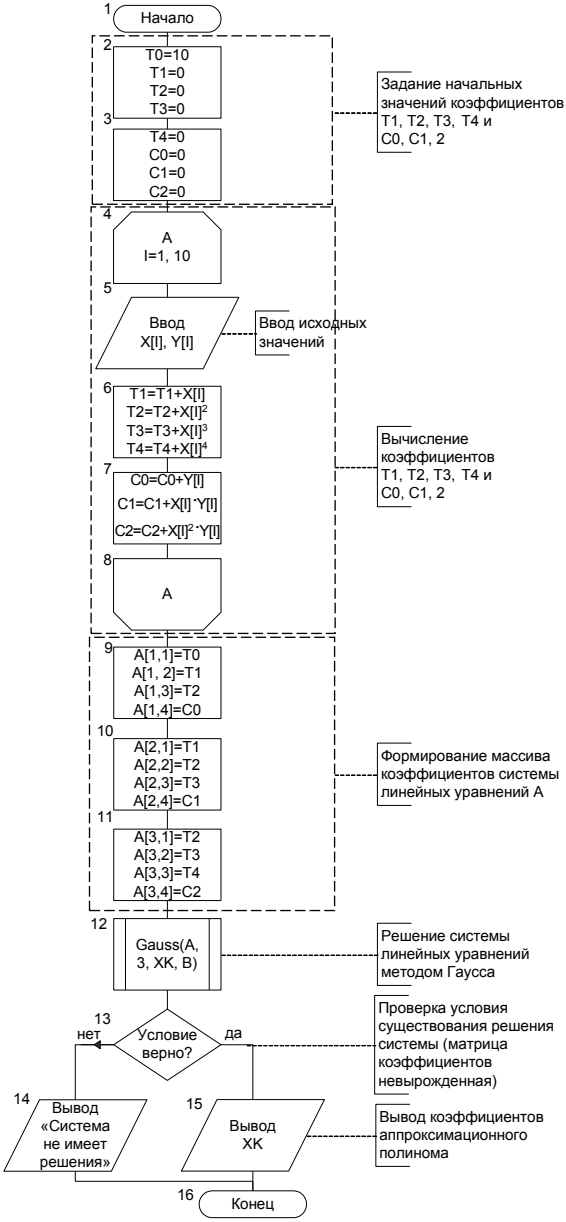

1
Начало
2
T0=10
T1=0
T2=0
T3=0
3
T4=0
C0=0
C1=0
C2=0
Задание начальных значений коэффициентов T1, T2, T3, T4 и C0, C1, 2
4
A
I=1, 10
5
Ввод X[I], Y[I]
Ввод исходных значений
6
T1=T1+X[I]
T2=T2+X[I]2
T3=T3+X[I]3
T4=T4+X[I]4
7
C0=C0+Y[I]
C1=C1+X[I]*Y[I]
C2=C2+X[I]2*Y[I]
8
A
Вычисление коэффициентов T1, T2, T3, T4 и C0, C1, 2
9
A[1,1]=T0
A[1, 2]=T1
A[1,3]=T2
A[1,4]=C0
10
A[2,1]=T1
A[2,2]=T2
A[2,3]=T3
A[2,4]=C1
11
A[3,1]=T2
A[3,2]=T3
A[3,3]=T4
A[3,4]=C2
Формирование массива коэффициентов системы линейных уравнений A
12
Gauss(A, 3, XK, B)
Решение системы линейных уравнений методом Гаусса
13
нет
Условие верно?
да
Проверка условия существования решения системы (матрица коэффициентов невырожденная)
14
Вывод «Система не имеет решения»
15
Вывод XK
Вывод коэффициентов аппроксимационного полинома
16
Конец

Рис. 3.7

3.4. **Дифференцирование и интегрирование табулированных функций**

***Задача** 3.8.* Разработать схему алгоритма и программу вычисления и вывода на экран значения интеграла $\int_{x_0}^{x_1}(12x^2+e^x)dx$ с помощью метода прямоугольников с шагом интегрирования h. Значения h, x_0, x_1 задаются с клавиатуры. Вычисление значения функции $12x^2+e^x$ оформить в виде подпрограммы (*function*).

Решение задачи

Схема алгоритма решения задачи представлена на рис. 3.8.

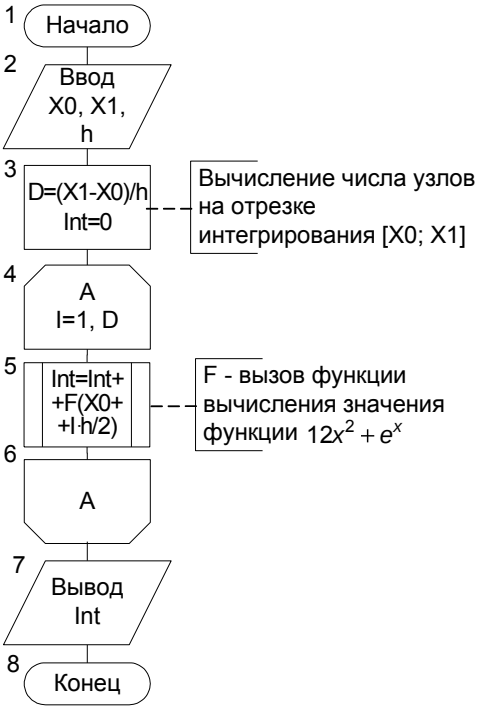


Рис. 3.8

Пояснения к схеме алгоритма

Обозначения:

Int – значение интеграла, вычисленное методом прямоугольников;

F – функция для вычисления значения функции $12x^2 + e^x$;

D – число узлов на отрезке интегрирования.

Символ 1. Начало алгоритма.

Символ 2. Ввод начальных данных.

Символ 3. Вычисление числа разбиений на прямоугольники (D) – числа узлов на отрезке интегрирования.

Символы 4–6. Вычисление в цикле с параметром *I* = 1; *D* суммы *Int* с помощью функции *F* для вычисления исходной функции.

Символ 7. Вывод на экран значения интеграла.

Символ 8. Конец алгоритма.

Программа

```
Program Intgr;
Var
   X0, X1, h, Int : Real;
   D, I : Integer;
Function F(X : Real) : real;
Begin          {начало раздела операторов функции F}
   F := 12*X*X + Exp(X);
end;           {конец функции F}
Begin          {начало раздела операторов основной программы}
   {ввод пределов интегрирования X0, X1}
   Write('Введите пределы интегрирования '); Readln(X0, X1);
   Write('Введите шаг h '); Readln(h); {ввод шага h}
   D:=Round((X1-X0)/h);    {вычисление числа узлов отрезка}
   Int := 0;          {первонач. значение значения интеграла}
   For I:=1 to D do   {цикл – по узлам отрезка интегрирования}
     Int := Int + F(X0 + I*h/2)*h; {вычисление интеграла}
   Writeln('Значение интеграла = ', Int:6:2); {вывод результата}
End.               {конец основной программы}
```

Результаты работы программы

```
Введите пределы интегрирования 1 5
Введите шаг h    0.1
Значение интеграла = 248.43
```

***Задача** 3.9.* Разработать алгоритм и программу вычисления двойного интеграла $\int_0^1 dx \int_2^3 (xy - 4x^3y^3)dy$ с помощью метода прямоугольников с шагом интегрирования h_1 по переменной x и с шагом интегрирования h_2 по переменной y. Шаги интегрирования h_1 и h_2 задаются с клавиатуры. Вычисление выражения $f(x, y) = xy - 4x^3y^3$ оформить в виде функции. Полученный результат вывести на экран. Решить аналитически заданный интеграл и сравнить полученные результаты.

Решение задачи

Суть аналитического вычисления двойного интеграла сводится к последовательному вычислению двух определенных интегралов (внутреннего и внешнего). Во внутреннем интеграле (по переменной y) переменная x принимается при интегрировании за постоянную.

Для численного вычисления двойного интеграла необходимо использовать следующий алгоритм.

Шаг 1. Организация цикла для перебора значений переменной x с шагом h_1 (шаг интегрирования внешнего интеграла).

Шаг 2. Для каждого значения x определение значения внутреннего интеграла методом прямоугольников (используется технология предыдущей задачи).

Шаг 3. Суммирование произведений значений внутренних интегралов и шага h_2.

Результат аналитического решения задачи: $\int_0^1 dx \int_2^3 (xy - 4x^3y^3)dy = -15$. Схема алгоритма численного решения задачи представлена на рис. 3.9.

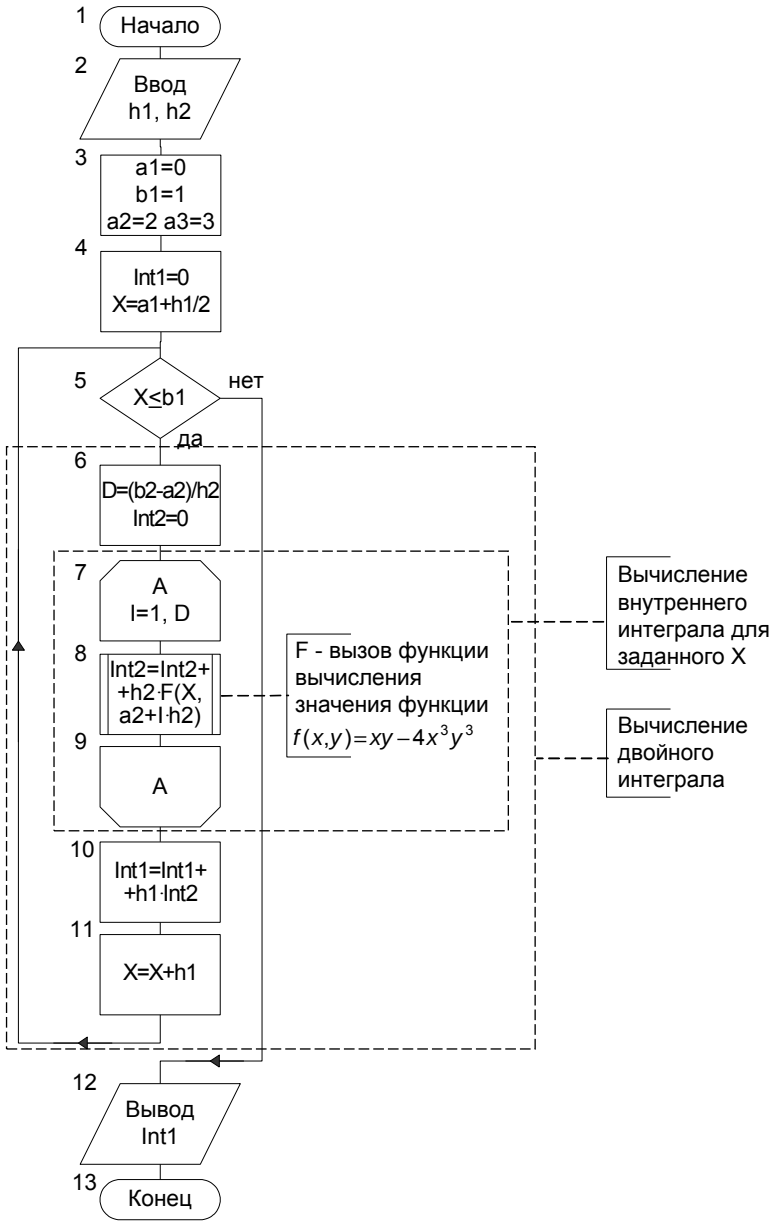

1
Начало
2
Ввод
h1, h2
3
a1=0
b1=1
a2=2 a3=3
4
Int1=0
X=a1+h1/2
5
X≤b1
нет
да
6
D=(b2-a2)/h2
Int2=0
7
A
I=1, D
8
Int2=Int2+
+h2·F(X,
a2+I·h2)
9
A
10
Int1=Int1+
+h1·Int2
11
X=X+h1
12
Вывод
Int1
13
Конец
F - вызов функции вычисления значения функции
$f(x,y)=xy-4x^3y^3$
Вычисление внутреннего интеграла для заданного X
Вычисление двойного интеграла

Рис. 3.9

Пояснения к схеме алгоритма

Обозначения:

*a*1, *b*1 – нижний и верхний пределы для внешнего интеграла;

*a*2, *b*2 – нижний и верхний пределы для внутреннего интеграла;

*Int*1 – значение двойного интеграла;

*Int*2 – значение внутреннего интеграла, вычисленное методом прямоугольников;

D – число узлов интегрирования по переменной *x*.

Символ 1. Начало алгоритма.

Символ 2. Ввод значений шагов интегрирования *h*1, *h*2.

Символ 3. Присвоение значений для пределов интегрирования внутреннего и внешнего интегралов.

Символ 4. Обнуление переменной суммы *Int*1 и вычисление начального значения переменной *X*.

Символ 5. Проверка условия $X \le b1$. Если условие верно, то выполняется *символ 6*, иначе – *символ 12*.

Символ 6. Вычисление числа интервалов для внутреннего интеграла. Обнуление переменной суммы внутреннего интеграла *Int*2.

Символ 7. Открытие цикла с параметром $I = 1$; *D*.

Символ 8. Вычисление переменной суммы двойного интеграла *Int*2 с использованием функции *F*.

Символ 9. Закрытие цикла с параметром *I*.

Символ 10. Вычисление переменной суммы двойного интеграла *Int*1.

Символ 11. Увеличение значение параметра *X* на *h*1.

Символ 12. Вывод на экран значения *Int*1.

Символ 13. Конец алгоритма.

Программа

```
Program Intgr2;
Var
   h1, h2, a1, a2, b1, b2, X, Int1, Int2 : Real;
   D, I : Integer;
Function F(X, Y : Real) : Real;
Begin          {начало раздела операторов функции F}
   F := X*Y - 4*X*X*X*Y*Y*Y;
end;                  {конец функции F}
```

```
Begin          {начало раздела операторов основной программы}
   Write('Введите значения шагов интегрирования: ');
   Readln(h1, h2);  {чтение значений шагов}
   Int1 := 0;
   a1 := 0; b1 := 1;{пределы интегрирования внешнего интеграла}
   a2 := 2; b2 := 3; {пределы интегрирования внутреннего интеграла}
   X := a1 + h1/2;
   While X <= b1 do
   begin
     D := Round((b2 - a2)/h2); {число узлов интегрирования}
     {вычисление внутреннего интеграла}
     Int2 := 0;
     for I :=  1 to D do Int2 := Int2 + F(X, a2+I*h2)*h2;
     Int1 := Int1+Int2*h1; {суммирование для двойного интеграла}
     X := X + h1;
   end;
   {вывод результата}
   Writeln('Значение двойного интеграла = ', Int1:6:2);
End.           {конец основной программы}
```

Результаты работы программы

```
Введите значения шагов интегрирования: 0.1  0.1
Значение двойного интеграла = -15.85

Введите значения шагов интегрирования: 0.001  0.001
Значение двойного интеграла = -15.01
```

Задача 3.10. Разработать схему алгоритма и программу вычисления первой и второй производной функции $f(x)=12x^2+e^x$ по двум узлам для точек x_0 и x_1. Значение первой производной вычислить, используя формулу центральных разностей (П 2.11), и оформить в виде функции. Значение второй производной вычислить с использованием конечно-разностной аппроксимации первой производной (П 2.15). Шаг сетки дифференцирования hp, значения x_0 и x_1 задаются с клавиатуры.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 3.10.

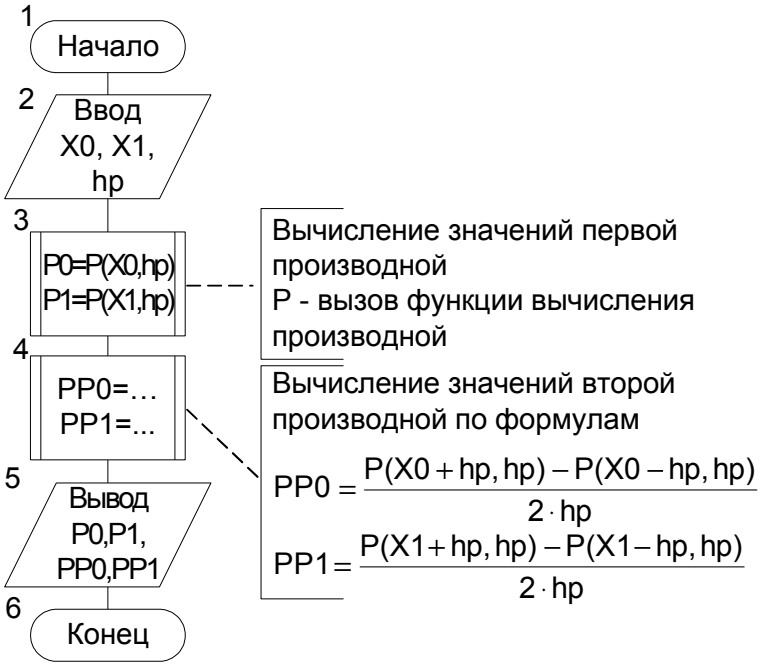


Рис. 3.10

Пояснения к схеме алгоритма

Обозначения:

P – функция для вычисления первой производной функции *f*(*x*);

*P*0, *P*1 – значения первой производной для точек x_0 и x_1;

*PP*0, *PP*1 – значения второй производной для точек x_0 и x_1;

hp – шаг сетки дифференцирования.

Символ 1. Начало алгоритма.

Символ 2. Ввод начальных данных.

Символ 3. Вычисление значений первой производной *P*0 в точке *X*0 и *P*1 в точке *X*1 с помощью вызова функции *P*.

Символ 4. Вычисление значений второй производной *PP*0 в точке *X*0 и *PP*1 в точке *X*1 с помощью вызова функции *P*.

Символ 5. Вывод на экран значений производных.

Символ 6. Конец алгоритма.

Программа

```
Program Proizv;
Var
   X0, X1, hp, P0, P1, PP0, PP1 : Real;
Function F(X : Real) : Real;
Begin                {начало раздела операторов функции F}
   F := 12*X*X + Exp(X);
end;                 {конец функции F}
Function P(X, hp : Real) : Real;
Begin                {начало раздела операторов функции P}
   P := (F(X + hp) - F(X - hp))/(2*hp);
end;                 {конец функции P}
Begin                {начало раздела операторов основной программы}
   {ввод значения узлов}
   Write('Введите значения узлов); Readln(X0, X1);
   Write('Введите шаг hp '); Readln(hp); {ввод шага hp}
   {вычисление значений первых производны в точках X0 и X1}
   P0:=P(X0, hp); P1:=P(X1, hp);
   {вычисление значения второй производной в точке X0}
   PP0 := (P(X0+hp,hp) - P(X0-hp,hp))/(2*hp);
   {вычисление значения второй производной в точке X1}
   PP1 := (P(X1+hp, hp) - P(X1-hp, hp))/(2*hp);
   {вывод результатов на экран}
   Writeln('Первая производная в точке X0 = ', P0:6:2, ' в
точке X1 = ', P1:6:2);
   Writeln('Вторая производная в точке X0 = ', PP0:6:2, ' в
точке X1 = ', PP1:6:2)
End.                 {конец основной программы}
```

Результаты работы программы

```
Введите значения узлов 1 5
Введите шаг hp   0.001
Первая производная в точке X0 = 26.72 в точке X1 = 268.41
Вторая производная в точке X0 = 26.72 в точке X1 = 172.42
```

3.5. Решение дифференциальных уравнений

3.5.1. Задача с использованием метода Эйлера

Задача 3.11. Разработать схему алгоритма и программу решения дифференциального уравнения $y' = 2y + e^x$ на отрезке [0, 2] мето-

дом Эйлера. Начальное условие Коши $y(0) = 1$. Вывести на экран решения $y(x)$ для различного числа разбиений отрезка для $n = 10$, 100, 1 000 и 10 000. Значение n вводить с помощью клавиатуры.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 3.11.

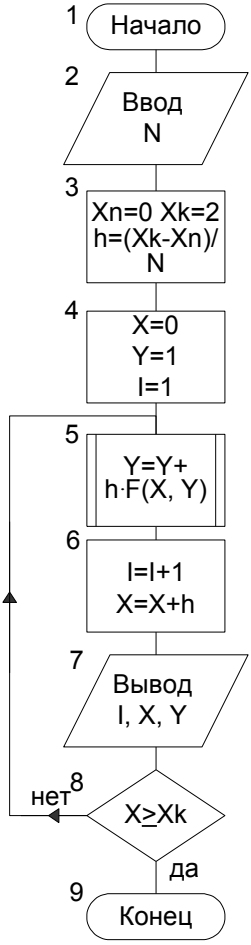


Рис. 3.11

Пояснения к схеме алгоритма

Обозначения:

Xn, *Xk* – начальное и конечное значение отрезка, на котором производится поиск решения;

F(*X*, *Y*) – функция для вычисления значения функции $2y+e^x$;

h – шаг изменения переменной *x*;

I – номер точки.

Символ 1. Начало алгоритма.

Символ 2. Ввод значения *N* – числа точек отрезка.

Символ 3. Задание границ отрезка поиска корня – [*Xn*, *Xk*]. Вычисление значение приращения для переменной $X - h$.

Символ 4. Задание значений *X*, *Y* для первой точки отрезка – начальное условие Коши. Переменной номера точки (*I*) присваивается значение 1.

Символ 5. Вычисление *Y* для следующей точки с использованием значений *X*, *Y* предыдущей точки. Значение уравнения $y' = 2y+e^x$ вычисляется с помощью функции *F*.

Символ 6. Значение номера точки *I* увеличивается на 1. Значение переменной *X* увеличивается на величину шага *h*.

Символ 7. Вывод на экран значений *I*, *X*, *Y*.

Символ 8. Проверка условия $X \geq Xk$. Если условие верно, то выполняется *символ 9*, если нет – *символ 5*.

Символ 9. Конец алгоритма.

Программа

```
Program Ejler;
Var
   Xn, Xk, h, X, Y, YF : Real;
   I, N : Integer;
Function F(X, Y : Real) : Real;
Begin          {начало раздела операторов функции F}
   F := 2*Y + Exp(X);
end;           {конец функции F}
Begin          {начало раздела операторов основной программы}
   Xn := 0;  Xk := 2;       {границы отрезка}
   {чтение значения n}
   Write('Введите число точек отрезка N '); Readln(N);
```

```
   h := (Xk - Xn)/N;        {приращение для переменной X}
   X := 0; Y := 1;          {начальное условие Коши}
   I := 0;           {первоначальное значение номера точки}
   Repeat            {начало цикла}
     {вычисление Y с использованием X, Y предыдущей точки}
     Y := Y + h*F(X, Y);
     I := I + 1;     {номер точки увеличивается на 1}
     X := X + h;     {значение X для I-й точки}
     {вывод результатов итерации}
     Writeln(I, ' точка X = ', X:4:2, ' Y = ', Y:7:4);
   Until X >= Xk; {выход из цикла, когда X станет больше Xk}
End.               {конец основной программы}
```

Результаты работы программы

```
Введите число точек отрезка 10
1 точка X = 0.2000 Y = 1.6000
2 точка X = 0.4000 Y = 2.4843
3 точка X = 0.6000 Y = 3.7764
4 точка X = 0.8000 Y = 5.6513
5 точка X = 1.0000 Y = 8.3570
6 точка X = 1.2000 Y = 12.2434
7 точка X = 1.4000 Y = 17.8048
8 точка X = 1.6000 Y = 25.7377
9 точка X = 1.8000 Y = 37.0234
10 точка X = 2.0000 Y = 53.0428
```

```
Введите число точек отрезка 100
1 точка X =  0.0200 Y = 1.0600
2 точка X =  0.0400 Y = 1.1228
3 точка X =  0.0600 Y = 1.1885
4 точка X =  0.0800 Y = 1.2573
…
96 точка X = 1.9200 Y = 79.8924
97 точка X = 1.9400 Y = 83.2245
98 точка X = 1.9600 Y = 86.6927
99 точка X = 1.9800 Y = 90.3024
100 точка X = 2.0000 Y = 94.0593
```

```
Введите число точек отрезка 1000
1 точка X = 0.0020 Y = 1.0060
2 точка X = 0.0040 Y = 1.0120
3 точка X = 0.0060 Y = 1.0181
4 точка X = 0.0080 Y = 1.0242
…
996 точка X = 1.9920 Y = 99.3283
997 точка X = 1.9940 Y = 99.7403
998 точка X = 1.9960 Y = 100.1540
999 точка X = 1.9980 Y = 100.5693
1000 точка X = 2.0000 Y = 100.9863
```

```
Введите число точек отрезка 10000
1 точка X = 0.0002 Y = 1.0006
2 точка X = 0.0004 Y = 1.0012
3 точка X = 0.0006 Y = 1.0018
4 точка X = 0.0008 Y = 1.0024
…
9996 точка X = 1.9992 Y = 101.5562
9997 точка X = 1.9994 Y = 101.5983
9998 точка X = 1.9996 Y = 101.6404
9999 точка X = 1.9998 Y = 101.6825
10000 точка X = 2.0000 Y = 101.7247
```

3.5.2. Задача с использованием метода Рунге-Кутта

***Задача** 3.12.* Разработать схему алгоритма и программу решения дифференциального уравнения $y' = 2y + e^x$ на отрезке [0, 2] методом Рунге-Кутта четвертого порядка. Начальное условие Коши $y(0) = 1$. Вывести на экран значения x и y в 10 и 1000 точках отрезка. Значение числа точек вводить с помощью клавиатуры.

Решение задачи

Схема алгоритма решения задачи представлена на рис. 3.12. Согласно этой схеме ниже представлен текст программы.

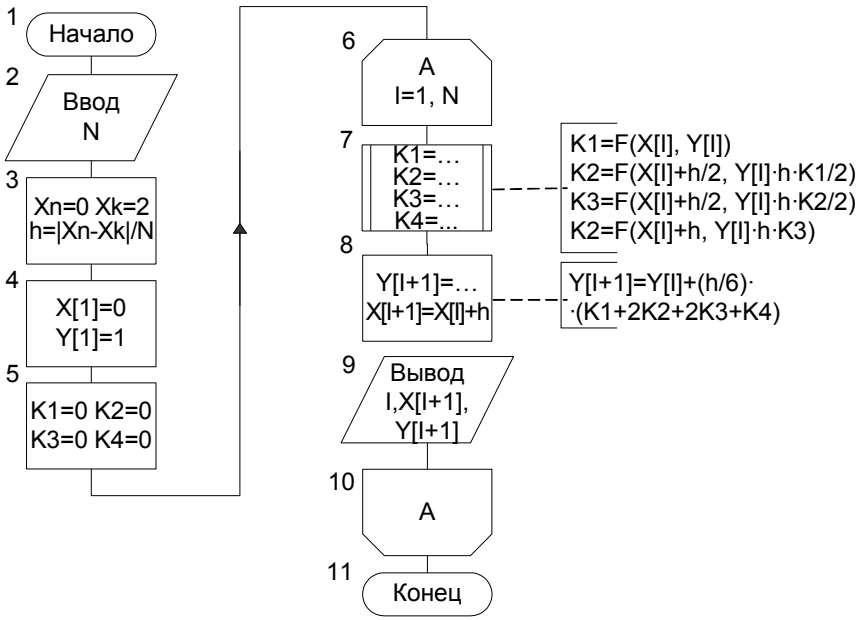


Рис. 3.12

Пояснения к схеме алгоритма

Обозначения:

Xn, *Xk* – начальное и конечное значения отрезка, на котором производится поиск решения;

h – шаг изменения переменной *x*;

Y[*I* + 1] – y_{i+1} элемент массива *Y* (*Y* – массив значений функции *y*);

X[*I*], *X*[*I* + 1] – x_i, $_{i+1}$ элементы массива *X* (*X* – массив значений функции *x*);

*K*1, *K*2, *K*3, *K*4 – коэффициенты метода Рунге-Кутта;

I – номер точки.

Символ 1. Начало алгоритма.

Символ 2. Ввод значения N – числа точек отрезка.

Символ 3. Задание границ отрезка поиска корня – [Xn, Xk]. Вычисление значение приращения h для переменной X.

Символ 4. Задание значений $X[1]$, $Y[1]$ для первой точки отрезка – начальное условие Коши.

Символ 5. Присвоение коэффициентам $K1$, $K2$, $K3$, $K4$ начальных значений.

Символ 6. Открытие цикла с параметром $I = 1$; N для перебора точек отрезка.

Символ 7. Вычисление значений коэффициентов $K1$, $K2$, $K3$, $K4$. Значение уравнения $y' = 2y + e^x$ вычисляется с помощью функции F.

Символ 8. Вычисление значения $Y[I + 1]$ для точки $(I + 1)$ с использованием значений $Y[I]$ и коэффициентов $K1$–$K4$. Присвоение значения переменной $X[I + 1]$.

Символ 9. Вывод на экран значений I, $X[I + 1]$, $Y[I + 1]$.

Символ 10. Закрытие цикла с параметром I.

Символ 11. Конец алгоритма.

Программа

```
Program Runge_Cutta;
Uses Crt;
Var
   Y, X : Array [0..1001] of Real;
   K1, K2, K3, K4, h, Xn, Xk : Real;
   I, N : Integer;
{функция вычисления значения f(x, y)}
Function F(X, Y : Real) : Real;
Begin         {начало раздела операторов функции F}
   F:= 2*Y + Exp(X);
end;                  {конец функции F}
Begin          {начало раздела операторов основной программы}
   ClrScr;            {очистка экрана}
   Xn:= 0;  Xk:= 2; {границы отрезка}
   {чтение значения N}
   Write('Введите число точек отрезка N '); Readln(N);
   h:= Abs(Xn - Xk)/N;      {вычисление шага изменения X}
   X[0]:= 0; Y[0]:= 1;      {начальное условие Коши}
   K1:=0; K2:=0; K3:=0; K4:=0;{начальные значения коэффициентов}
   for I := 0 to N - 1 do
   Begin                    {начало цикла 1}
```

```
      {вычисление коэффициентов K1, K2, K3, K4}
      K1:= F(X[I], Y[I]); K2:= F(X[I]+h/2, Y[I]+h*K1/2);
      K3:= F(X[I]+h/2, Y[I]+h*K2/2);
      K4:= F(X[I]+h, Y[I]+h*K3);
      {вычисление Y в (I+1)-й точке}
      Y[I+1]:=Y[I]+(h/6)*(K1+2*K2+2*K3+K4);
      X[I+1]:= X[I] + h;    {значение X в (I+1)-й точке}
      Writeln(I, ' точка X = ', X[I+1]:2:1, ' Y = ', Y[I+1]:5:3);
   End;                     {конец цикла 1}
   Repeat Until KeyPressed;{ожидание нажатия любой клавиши}
end.                        {конец основной программы}
```

Результаты работы программы

```
Введите число точек отрезка 10
1 точка X = 0.2 Y = 1.762
2 точка X = 0.4 Y = 2.959
3 точка X = 0.6 Y = 4.817
4 точка X = 0.8 Y = 7.678
5 точка X = 1.0 Y = 12.056
6 точка X = 1.2 Y = 18.719
7 точка X = 1.4 Y = 28.821
8 точка X = 1.6 Y = 44.090
9 точка X = 1.8 Y = 67.109
10 точка  X = 2.0 Y = 101.745
```

```
Введите число точек отрезка 1000
1 точка X = 0.0 Y = 1.006
2 точка X = 0.0 Y = 1.012
3 точка X = 0.0 Y = 1.018
4 точка X = 0.0 Y = 1.024
…
996 точка X = 2.0 Y = 100.133
997 точка X = 2.0 Y = 100.549
998 точка X = 2.0 Y = 100.967
999 точка X = 2.0 Y = 101.386
1000 точка X = 2.0 Y = 101.807
```

3.5.3. Задача с использованием метода конечных разностей

***Задача** 3.13.* Разработать схему алгоритма и программу решения двумерного уравнения теплопроводности конечно-разностным методом для заданной области, используя следующие параметры для типов клеток:

– для типа 1: теплоемкость c = 519 Дж/К, теплопроводность λ = 44,4 Вт/(м·К), плотность ρ = 1600 кг/м^3, начальная температура t = 100 °С;

– для типа 2: теплоемкость c = 343 Дж/К, теплопроводность λ = 110 Вт/(м·К), плотность ρ = 8900 кг/м^3, начальная температура t = 1100 °С;

С шагами по пространству $\Delta x = \Delta y$ = 0,002 м и по времени Δt = 0,00005 с рассчитать температурное поле до времени 0,1 с.

1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 2 2 2 1 1 1 1
1 1 1 1 2 2 2 1 1 1 1
1 1 1 1 2 2 2 1 1 1 1
1 1 1 2 2 2 2 2 1 1 1
1 1 1 2 2 2 2 2 1 1 1
1 1 1 2 2 2 2 2 1 1 1
1 1 1 1 2 2 2 1 1 1 1
1 1 1 1 2 2 2 1 1 1 1
1 1 1 1 2 2 2 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1

Решение

Учитывая, что шаги по осям x и y, Δx и Δy равны, равенство (П 2.26) будет иметь вид

$$T_{i,j}^{n+1} = T_{i,j}^{n} + a\Delta t\left(\frac{T_{i+1,j}^{n} + T_{i-1,j}^{n} + T_{i,j+1}^{n} + T_{i,j-1}^{n} - 4T_{i,j}^{n}}{\Delta x^2}\right), \qquad (3.1)$$

где $T_{i,j}$ – температура в точке с координатами i, j по осям x, y соответственно;

n, $n+1$ – координаты по времени;

$$a = \frac{\lambda}{c \cdot \rho},$$

где λ – теплопроводность;

c – теплоемкость;

ρ – плотность.

Схема алгоритма решения задачи представлена на рис. 3.13. Согласно этой схеме ниже представлен текст программы. Алгоритм содержит цикл по времени t с вложенными в него циклами по i и j, в которых перебираются сеточные элементы объекта, в каждом из которых определяются температуры в момент времени $\tau + \Delta\tau$.

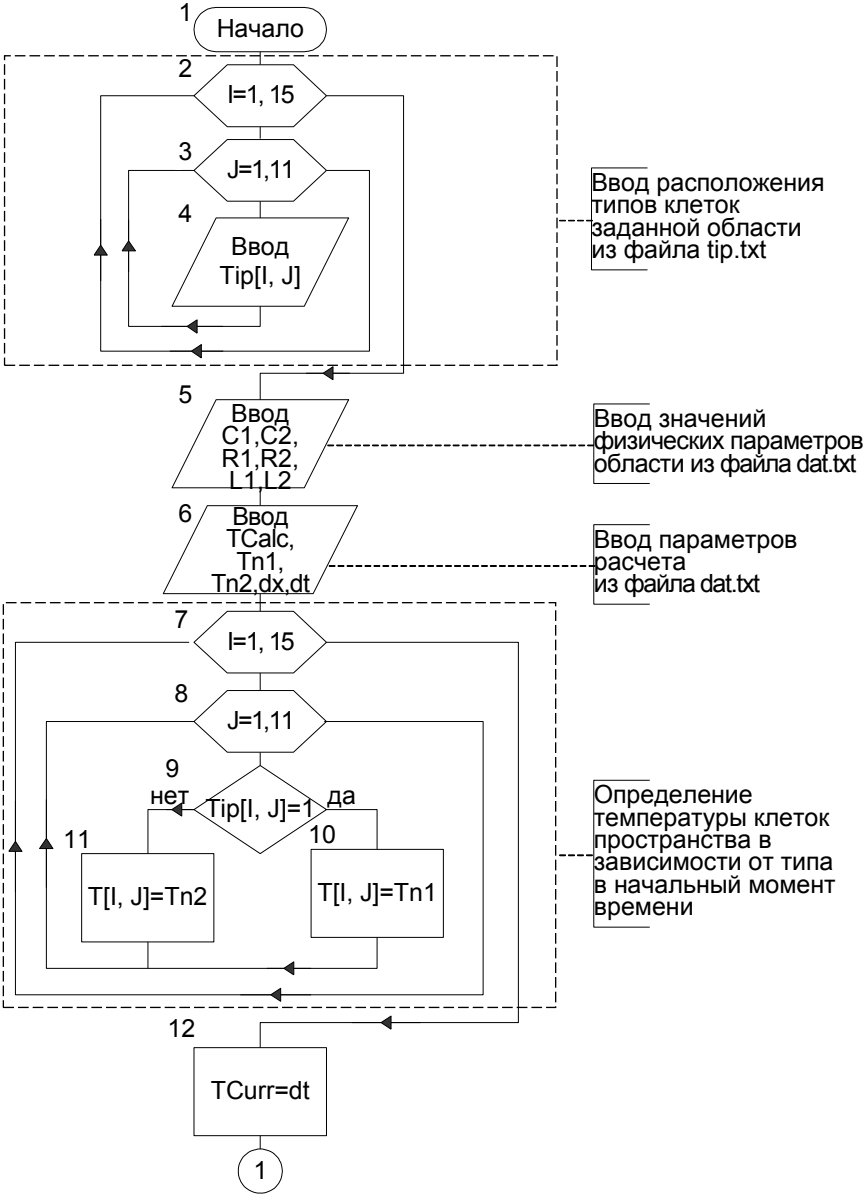

1
Начало
2
I=1, 15
3
J=1,11
4
Ввод
Tip[I, J]
Ввод расположения типов клеток заданной области из файла tip.txt
5
Ввод
C1,C2,
R1,R2,
L1,L2
Ввод значений физических параметров области из файла dat.txt
6
Ввод
TCalc,
Tn1,
Tn2,dx,dt
Ввод параметров расчета из файла dat.txt
7
I=1, 15
8
J=1,11
9
нет
Tip[I, J]=1
да
11
T[I, J]=Tn2
10
T[I, J]=Tn1
Определение температуры клеток пространства в зависимости от типа в начальный момент времени
12
TCurr=dt
1

Рис. 3.13

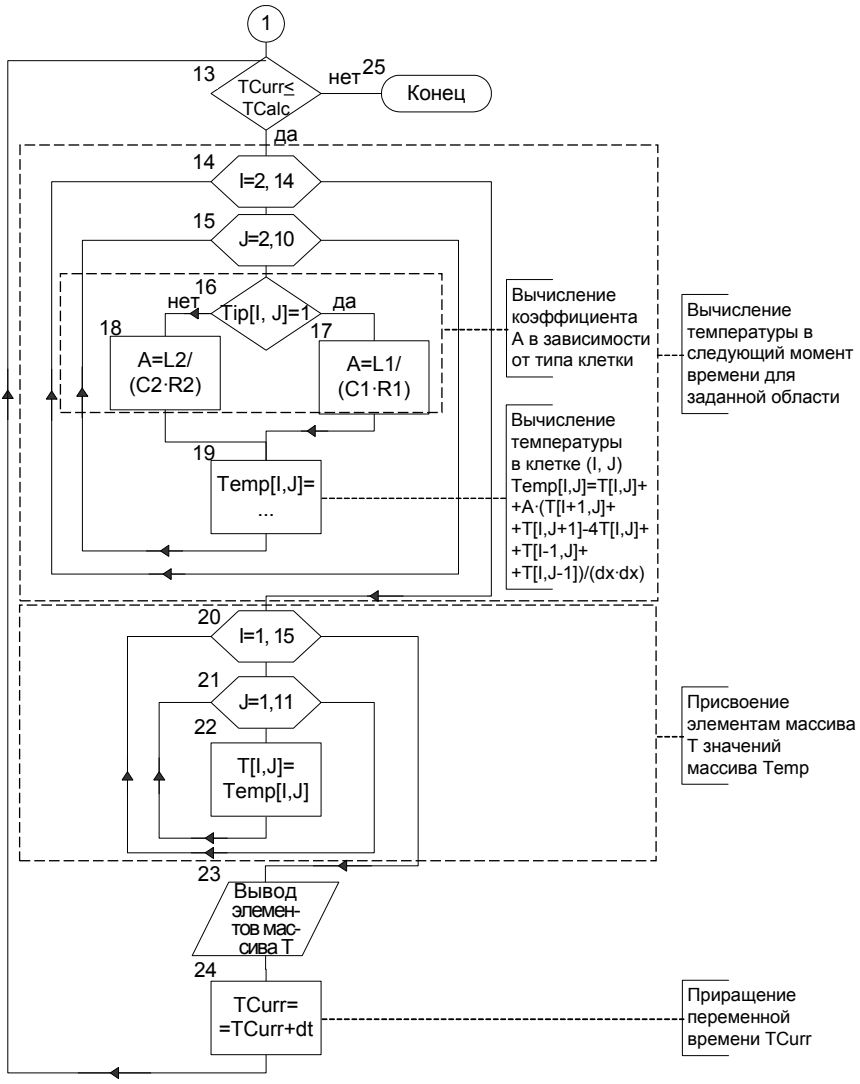


Рис. 3.13 (окончание)

Пояснения к схеме алгоритма

Обозначения:

Tip – массив размером 15 × 11, содержащий значения типов клеток заданной области, содержит значение 1 – если клетка типа 1, значение 2 – для клетки типа 2, заполняется значениями из файла *tip.txt*;

Tip[*I, J*] – элемент, расположенный в *i*-й строке и *j*-м столбце массива *Tip*;

T – массив размером 15 × 11 значений температур клеток заданной области.

T[*I, J*] – t_{ij} элемент массива *T*;

Temp – массив размером 15 × 11 для временного хранения температур клеток заданной области;

Temp[*I, J*] – элемент, расположенный в *i*-й строке и *j*-м столбце массива *Temp*;

*C*1, *C*2 – значения теплоемкости материалов типов 1 и 2;

*R*1, *R*2 – значения плотности материалов типов 1 и 2;

*L*1, *L*2 – значения теплопроводности материалов типов 1 и 2;

dx, *dt* – значения шагов изменения по пространству и времени;

*Tn*1, *Tn*2 – значения начальных температур клеток для типов 1 и 2 клеток заданной области;

TCurr – текущее время расчета;

TCalc – конечное время расчета.

Символ 1. Начало алгоритма.

Символ 2. Открытие цикла с параметром *I* = 1; 15 для перебора координат по оси *y* заданной области (перебор строк области).

Символ 3. Открытие цикла с параметром *J* = 1; 11 для перебора координат по оси *x* заданной области (перебор столбцов области).

Символ 4. Ввод типа клетки с координатами (*I*, *J*) из файла *tip.txt*.

Символ 5. Ввод значений физических параметров материалов типов 1 и 2: теплоемкостей (*C*1, *C*2), плотностей (*R*1, *R*2), теплопроводностей (*L*1, *L*2) из файла *dat.txt*.

Символ 6. Ввод значений параметров расчета: время расчета (*TCalc*), значения начальных температур для материалов типов 1 и 2 (*Tn*1, *Tn*2), шаги по пространству (*dx*) и времени (*dt*) из файла *dat.txt*.

Символ 7. Открытие цикла с параметром *I* = 1; 15 для перебора координат по оси *y* заданной области (перебор строк области).

Символ 8. Открытие цикла с параметром *J* = 1; 11 для перебора координат по оси *x* заданной области (пребор столбцов области).

Символ 9. Вычисление значения температуры клетки с координатами (*I*, *J*) в начальный момент времени.

Символ 10. Присвоение элементу массива *T*[*I*, *J*] значения *Tn*1.

Символ 11. Присвоение элементу массива *T*[*I*, *J*] значения *Tn*2.

Символ 12. Присвоение переменной текущего времени расчета *TCurr* значения *dt*.

Символ 13. Проверка условия, что текущее время (*TCurr*) не превышает конечное время расчета (*TCalc*). Если условие верно, то выполняется *символ 14*, если нет – *символ 25*.

Символ 14. Открытие цикла с параметром *I* = 2; 14 для перебора координат по оси *y* заданной области (перебор строк области).

Символ 15. Открытие цикла с параметром *J* = 1; 10 для перебора координат по оси *x* заданной области (перебор столбцов области).

Символ 16. Определение типа материала клетки с координатами (*I*, *J*). Проверка условия равенства значения элемента *Tip*[*I*, *J*] единице. Если условие верно, то выполняется *символ 17*, если нет – *символ 18*.

Символ 17. Вычисление значения коэффициента *A* для типа 1 клетки с координатами (*I*, *J*).

Символ 18. Вычисление значения коэффициента *A* для типа 2 клетки с координатами (*I*, *J*).

Символ 19. Вычисление значения температуры в клетке с координатами (*I*, *J*) в момент времени *TCurr* по формуле (3.1).

Символ 20. Открытие цикла с параметром *I* = 1; 15 для перебора координат по оси *y* заданной области (перебор строк области).

Символ 21. Открытие цикла с параметром *J* = 1; 11 для перебора координат по оси *x* заданной области (перебор столбцов области).

Символ 22. Присвоение элементу массива *T*[*I*, *J*] значения элемента массива *Temp*[*I*, *J*].

Символ 23. Вывод элементов массива *T* на экран. Вывод значений производится с организацией внешнего цикла для перебора строк массива *T* и внутреннего цикла для перебора столбцов (табл. П 1.13), на схеме для упрощения это показано в одном символе.

Символ 24. Приращение переменной текущего времени расчета (*TCurr*) на значение шага по времени (*dt*).

Символ 25. Конец алгоритма.

Программа

```
Program PartDeriv;
Uses Crt;
Var
   f : Text;
   T, tip, Temp : Array [1..15, 1..11] of Real;
   TCurr, TCalc, A : Real;
   L1, L2, C1, C2, R1, R2, Tn1, Tn2, dx, dt : Real;
   I, J : Integer;
Begin                {начало раздела операторов программы}
   ClrScr;           {очистка экрана}
   Assign(f,'tip.txt'); Reset(f); {открытие файла tip.txt}
   {чтение расположения типов клеток области из файла tip.txt}
   for I:=1 to 15 do
   Begin             {начало цикла 1}
     for J:=1 to 11 do Read(f, Tip[I, J]);
     Readln(f);      {переход на след. строку в файле}
   end;              {конец цикла 1}
   Close(f);         {закрытие файла tip.txt}
   Assign(f,'dat.txt'); Reset(f); {открытие файла dat.txt}
   {чтение физических параметров области из файла dat.txt}
   Read(f, C1);  Readln(f, C2);     {значения теплоемкостей}
   Read(f, R1);  Readln(f, R2);     {значения плотностей}
   Read(f, L1);  Readln(f, L2);     {значения теплопроводностей}
   {значения шагов по пространству и по времени}
   Read(f, dx);  Readln(f,dt);
   Readln(f, TCalc);       {значение времени расчета}
   Read(f, Tn1); Readln(f, Tn2);  {значения начальных температур}
   Close(f);  {закрытие файла dat.txt}
   {расчет}
   {температуры клеток области в начальный момент времени}
   for I:=1 to 15 do
      for J:=1 to 11 do
         If Tip[I, J]=1 Then T[I, J]:= Tn1 Else T[I, J]:=Tn2;
   TCurr:= dt;
   While TCurr <= TCalc do
   Begin                          {начало цикла 2}
     ClrScr;                      {очистка экрана}
     for I:=2 to 14 do
       for J:=2 to 10 do  {начало цикла 4}
       Begin                      {начало цикла 5}
         If Tip[I, J] = 1 Then A:= L1/(C1*R1)
                          Else A:= L2/(C2*R2);
         {температура в клетке (I,J) в след. момент времени}
         Temp[I, J]:=T[I, J]+A*dt*(T[I+1, J] + T[I, J+1] -
```

```
                    4*T[I, J] + T[I-1, J] + T[I, J-1])/(dx*dx);
      end;                  {конец циклов 4, 5}
    for I:=2 to 14 do
      for J:=2 to 10 do T[I, J]:= Temp[I, J];
    {вывод массива значений температур T на экран}
    for I:=1 to 15 do
    Begin                   {начало цикла 6}
      {вывод строки температур}
      for J:=1 to 11 do Write(' ',T[I, J]:6:1);
      Writeln(' ');         {переход на следующую строку}
    end;                    {конец цикла 6}
    Writeln('Время ', TCurr:7:5, ' Осталось итераций ',
Trunc((TCalc-TCurr)/dt):10);
    Writeln('Нажмите клавишу Enter');
    TCurr:= TCurr + dt;     {приращение переменной времени}
    Readln;                 {ожидание нажатия клавиши Enter}
  end;                      {конец цикла 2 - While}
End.                        {конец программы}
```

Результат работы программы

Файл dat.txt:

```
519 343
1600 8900
44.4 110
0.002 0.00005
0.1
100 1100
```

Поле температур после окончания расчета:

```
100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0
100.0 109.4 128.2 161.6 200.4 217.1 200.4 161.6 128.2 109.4 100.0
100.0 123.9 171.0 255.2 354.0 396.6 354.0 255.2 171.0 123.9 100.0
100.0 144.2 229.1 379.0 563.0 641.8 563.0 379.0 229.1 144.2 100.0
100.0 167.0 288.0 488.3 719.0 815.5 719.0 488.3 288.0 167.0 100.0
100.0 193.4 349.6 584.2 821.1 914.2 821.1 584.2 349.6 193.4 100.0
100.0 219.8 409.8 676.7 900.1 979.2 900.1 676.7 409.8 219.8 100.0
100.0 231.0 435.1 715.3 930.5 1002.7 930.5 715.3 435.1 231.0 100.0
100.0 219.8 409.8 676.7 900.1 979.2 900.1 676.7 409.8 219.8 100.0
100.0 193.4 349.6 584.2 821.1 914.2 821.1 584.2 349.6 193.4 100.0
100.0 167.0 288.0 488.3 719.0 815.5 719.0 488.3 288.0 167.0 100.0
100.0 144.2 229.1 379.0 563.0 641.8 563.0 379.0 229.1 144.2 100.0
100.0 123.9 171.0 255.2 354.0 396.6 354.0 255.2 171.0 123.9 100.0
100.0 109.4 128.2 161.6 200.4 217.1 200.4 161.6 128.2 109.4 100.0
100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0
```

4. ВЫЧИСЛИТЕЛЬНЫЕ ТЕХНОЛОГИИ РЕШЕНИЯ ЗАДАЧ В MS EXCEL

4.1. **Решение системы линейных уравнений матричным методом**

Задача 4.1. Решить систему линейных уравнений с помощью матричного метода. Систему линейных уравнений сформировать на основе заданного закона, определяющего коэффициенты при неизвестных. Принять, что число неизвестных системы равно четырем. Коэффициенты системы задаются по закону

$$a_{ij} = \begin{cases} 2\cos(i+j) \text{ при } i \le j; \\ 3\sin(i+j) \text{ при } i > j; \end{cases} \tag{4.1}$$

$$c_i = \operatorname{tg}(i+5) + i^2. \tag{4.2}$$

Решение задачи

Система уравнений с четырьмя неизвестными имеет вид

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = c_1, \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = c_2, \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = c_3, \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = c_4. \end{cases} \tag{4.3}$$

Используя уравнения (4.1) и (4.2), вычислим значения некоторых коэффициентов системы линейных уравнений:

a_{11} = 2cos(1 + 1) при 1 = 1;
a_{12} = 2cos(1 + 2) при 1 < 2;
a_{21} = 3sin(2 + 1) при 2 > 1;
c_1 = tg(1 + 5) + 1^2.

Алгоритм решения

Решим систему линейных уравнений (4.3) с помощью матричных операций. Для этого представим коэффициенты системы линейных уравнений (4.3) в виде матрицы коэффициентов $A_{4\times4}$, а свободные члены линейных уравнений (4.3) – в виде вектора $C_{4\times1}$. Корни системы уравнений (4.3) x_1, x_2, x_3, x_4 будут представлены как вектор неизвестных $X_{4\times1}$.

При формировании матрицы коэффициентов $A_{4\times4}$ и вектора свободных членов уравнений $C_{4\times1}$ можно использовать дополнительные матрицы $A1$ и $A2$. Дополнительные матрицы $A1$ и $A2$ формируются соответственно из индексов строк и столбцов матрицы A размером 4×4.

Шаг 1. Вначале сформируем матрицу $A_{4\times4}$, элементы которой найдем из соотношения (4.1) с помощью встроенной логической функции ЕСЛИ(), используя дополнительные матрицы $A1$ и $A2$. Для этого создадим дополнительные матрицы $A1$ и $A2$ и активизируем левую верхнюю ячейку из диапазона ячеек, который будет занимать матрица $A_{4\times4}$. Затем в строке меню выберем команду *Вставка функции* и в открывшемся окошке *Мастер функций* выберем логическую функцию ЕСЛИ() (рис. 4.1). Любую встроенную функцию можно выбрать в разделе *Категория* из списка *Полного алфавитного перечня.*

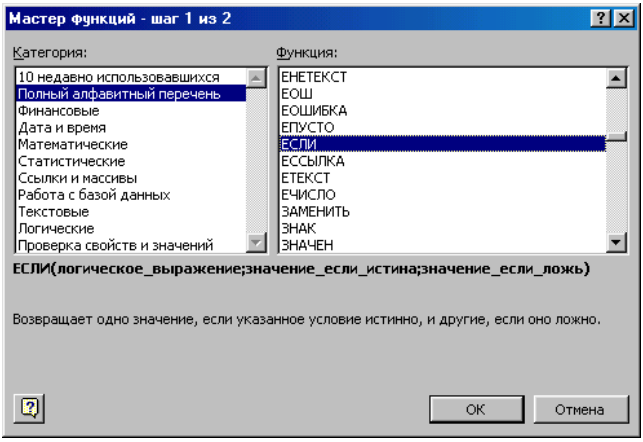


Рис. 4.1

Шаг 2. В диалоговом окне функции ЕСЛИ() заполним поля *Логическое_выражение*, *Значение_если_истина* и *Значение_если_ложь*. Запишем в них условие (4.1), используя адреса левых верхних ячеек дополнительных матриц $A1$ и $A2$ (рис. 4.2). Ввод формулы завершим, нажав клавишу *ОК*.

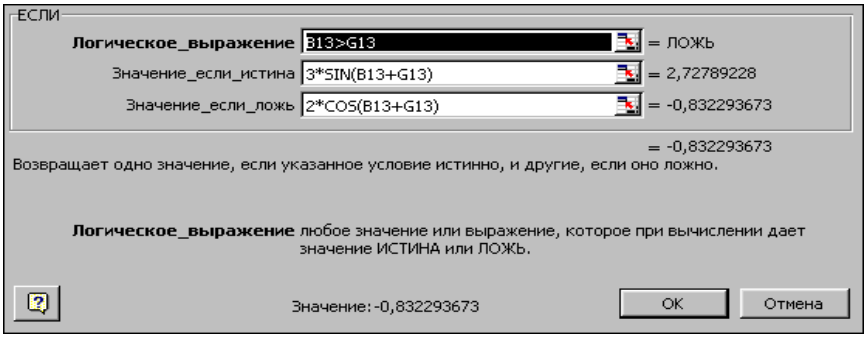
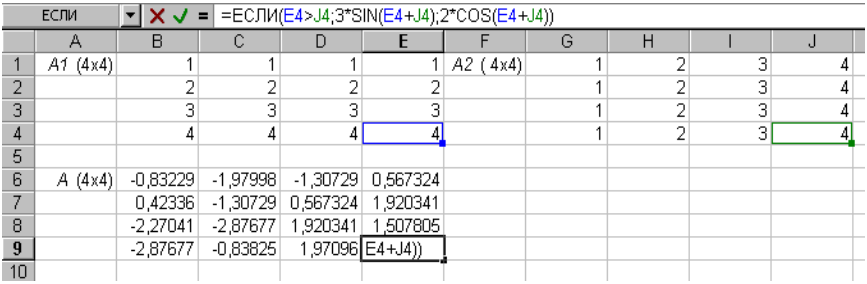


Рис. 4.2

Шаг 3. Скопируем полученное соотношение во все ячейки формируемой матрицы $A_{4\times4}$ или с помощью операции копирования, или перетаскивая маркер заполнения (рис. 4.3).



	A	B	C	D	E	F	G	H	I	J
1	A1 (4x4)	1	1	1	1	A2 (4x4)	1	2	3	4
2		2	2	2	2		1	2	3	4
3		3	3	3	3		1	2	3	4
4		4	4	4	4		1	2	3	4
5										
6	A (4x4)	-0,83229	-1,97998	-1,30729	0,567324					
7		0,42336	-1,30729	0,567324	1,920341					
8		-2,27041	-2,87677	1,920341	1,507805					
9		-2,87677	-0,83825	1,97096	E4+J4))					
10										

Рис. 4.3

Шаг 4. Сформируем вектор $C_{4\times1}$, элементы которого вычислим по соотношению (4.2) с помощью встроенных математических функций. Для этого в строке меню выберем команду *Вставка функции* и в открывшемся окошке *Мастер функций* выберем математическую функцию TAN() (рис. 4.4). В открывшемся диалоговом

окне функции TAN() в поле *Число* запишем аргумент тангенса (*i*+5) (рис. 4.5). В данном случае за *i* возьмем верхнюю ячейку первого столбца матрицы $A_{4\times4}$. Завершим ввод функции tg, нажав *OK*. Ещё раз зайдем в ячейку с формулой и дополним её слагаемым i^2, применив ссылку на верхнюю ячейку первого столбца матрицы $A_{4\times4}$. Скопируем полученное соотношение во все ячейки формируемой матрицы-вектора $C_{4\times1}$, используя данные первого столбца матрицы $A_{4\times4}$ (рис. 4.6).

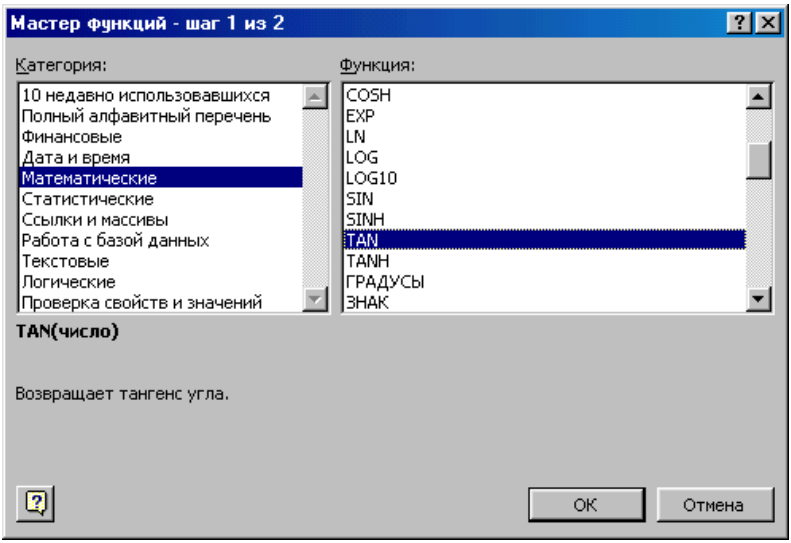


Рис. 4.4

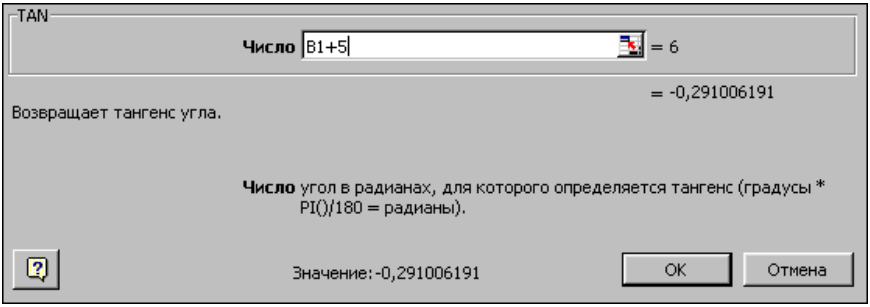


Рис. 4.5

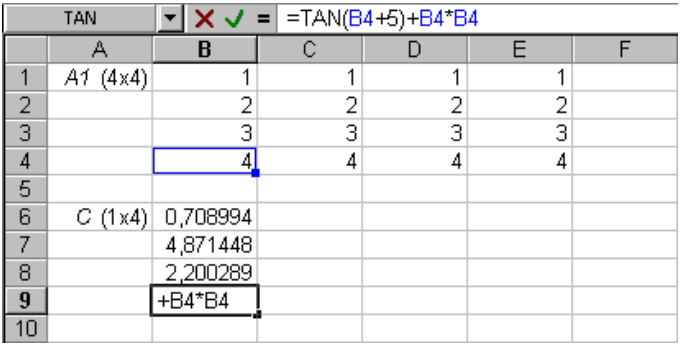


Рис. 4.6

Шаг 5. Систему уравнений (4.3) можно записать в матричном виде: $AX = C$. Чтобы найти вектор $X_{4\times1}$, необходимо обе части полученного уравнения умножить на матрицу $A^{-1}_{4\times4}$, обратную матрице $A_{4\times4}$: $AXA^{-1} = CA^{-1}$. Умножение матрицы A на обратную ей матрицу A^{-1} приводит к получению единичной матрицы: $X = CA^{-1}$. Чтобы найти матрицу $A^{-1}_{4\times4}$, используем функцию массива МОБР(). Для этого выделим группу ячеек, в которых должна разместиться обратная матрица (рис. 4.7), и активизируем последовательно команды *Вставка функции* → *МОБР* (рис. 4.8). Зайдя в диалоговое окно МОБР(), внесем координаты матрицы $A_{4\times4}$ в поле *Массив* (рис. 4.9). Для завершения действия с функцией МОБР() нажимаем одновременно комбинацию клавиш *Ctrl+Shift+Enter* (рис. 4.10).

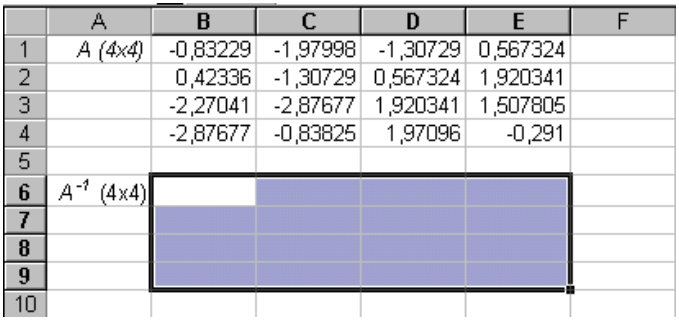


Рис. 4.7

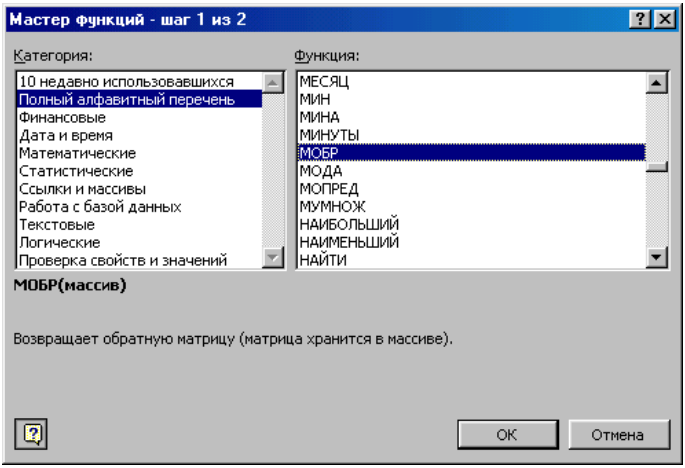


Рис. 4.8

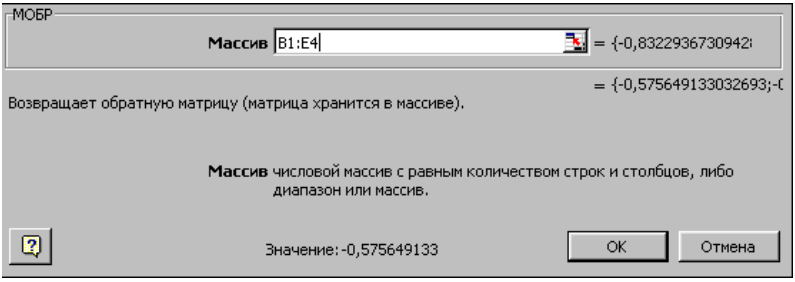


Рис. 4.9

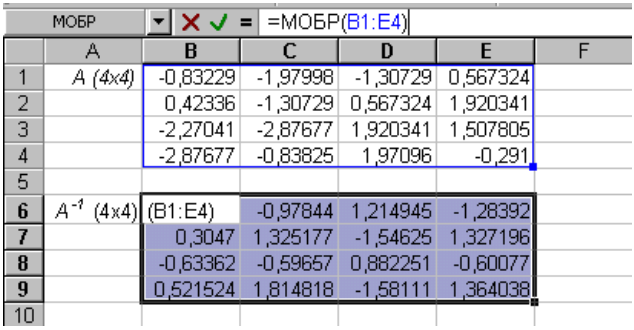


Рис. 4.10

Шаг 6. Найдем вектор неизвестных X, умножив обратную матрицу A^{-1} на вектор C: $X = CA^{-1}$. Для этого выделим группу ячеек, в которых должен разместиться вектор неизвестных (рис. 4.11), и активизируем последовательно команды *Вставка функции* → МУМНОЖ (рис. 4.12). В диалоговом окне функции МУМНОЖ с помощью блока выбора значений выделим последовательно матрицу A^{-1} и матрицу-вектор C и заполним поля *Массив1* и *Массив2* или просто запишем их координаты (рис. 4.13).

	A	B	C	D	E	F	G	H
1	A-1 (4x4)	-0,57565	-0,97844	1,214945	-1,28392	C (1x4)	0,708994	
2		0,3047	1,325177	-1,54625	1,327196		4,871448	
3		-0,63362	-0,59657	0,882251	-0,60077		2,200289	
4		0,521524	1,814818	-1,58111	1,364038		15,54768	
5								
6	X (1x4)	x_1						
7		x_2						
8		x_3						
9		x_4						

Рис. 4.11

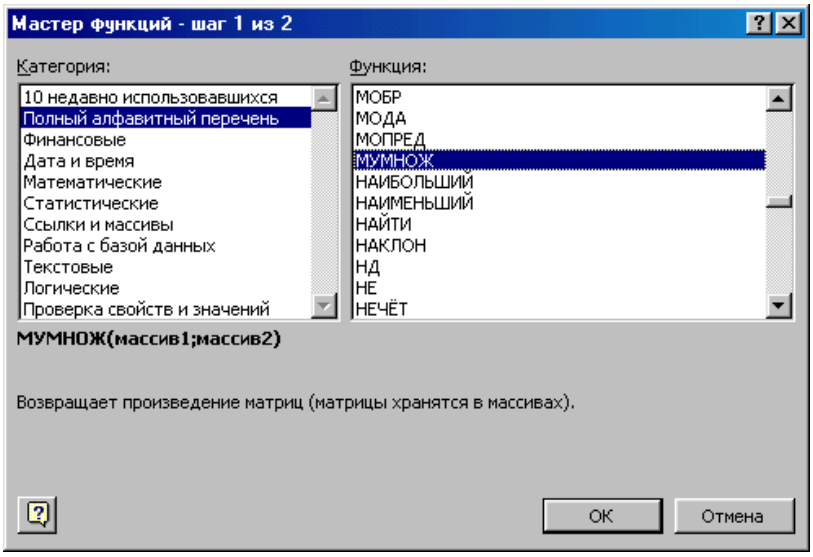


Рис. 4.12

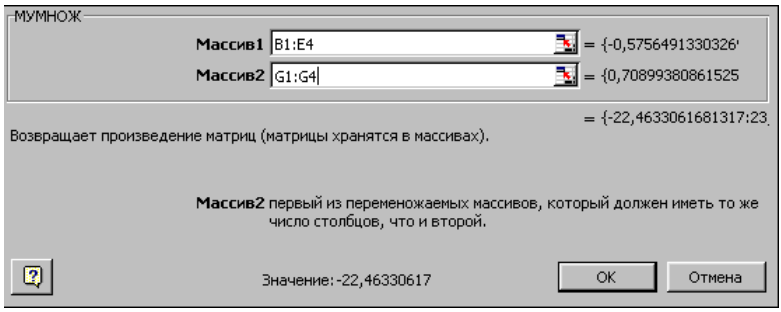


Рис. 4.13

Завершаем действие с функцией МУМНОЖ, нажав одновременно комбинацию клавиш *Ctrl+Shift+Enter* (рис. 4.14).

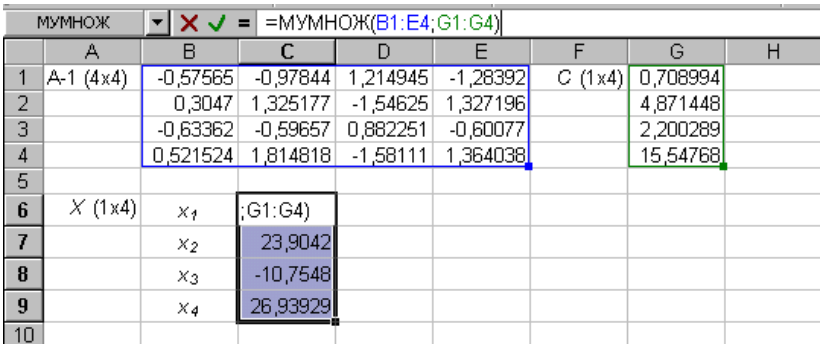


Рис. 4.14

В результате решения системы линейных уравнений (4.3) с помощью матричных операций получены значения неизвестных: $x_1 = -22{,}46$, $x_2 = 23{,}9$, $x_3 = -10{,}74$, $x_4 = 26{,}94$ (см. рис. 4.14).

Шаг 7. Проверим правильность решения системы линейных уравнений (4.3) с помощью подстановки, т. е. запишем каждое уравнение и системы в ячейки, указывая координаты ячеек с соответствующими значениями элементов матрицы $A_{4\times4}$ a_{11}, a_{12}, …, a_{44} и элементов матрицы-вектора $X_{4\times1}$ x_1, x_2, x_3, x_4 (рис. 4.15).

В результате проведенной проверки получен вектор свободных членов системы уравнений (4.3), аналогичный вектору $C_{4\times1}$, т. е. система уравнений (4.3) решена правильно (см. рис. 4.15).

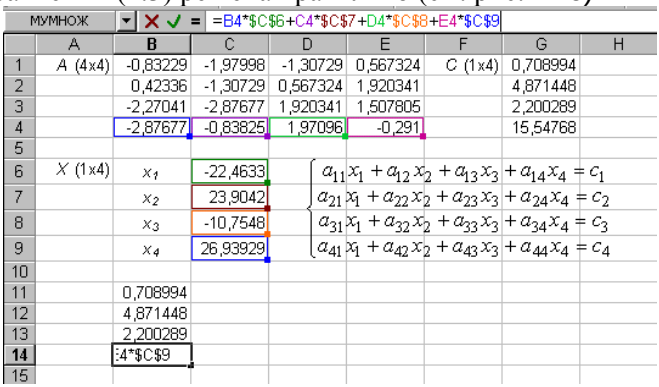


Рис. 4.15

4.2. **Решение нелинейных уравнений и систем**

4.2.1. Задача решения нелинейного уравнения методом дихотомии

***Задача** 4.2.* Решить нелинейное уравнение $f(x) = e^{x/2} - x^3 + 10$ на отрезке [1, 4] методом дихотомии с точностью 0,01. Проиллюстрировать графически положение корня в плоскости *XY*.

Алгоритм решения

Обозначим отрезок, на котором будет производиться поиск корня, как [*a*, *b*]. Суть метода состоит в циклическом выполнении двух операций:

1) уточнение отрезка существования корня уравнения (выбор половины отрезка [*a*, *b*], который содержит корень уравнения);

2) выбор нового отрезка до тех пор, пока допустимая погрешность вычисления корня не станет меньшей либо равной заданной точности (0,01).

Предварительное табулирование. Найдем значения функции $f(x) = e^{x/2} - x^3 + 10$ на отрезке [1, 4], используя шаг изменения переменной *x*, равный 0,2, т. е. проведем предварительное табулирование уравнения (рис. 4.16). Как видно из рисунка, заданная функция меняет знак на отрезке [1, 4] и, значит, можно найти корень функции *f*(*x*) на этом отрезке.

Построим график функции *f*(*x*) с помощью функции «*Мастер диаграмм*». Для этого выделим диапазон данных B4 : C19, активизируем функцию «*Мастер диаграмм*», выберем тип «*Точечная диаграмма со значениями, соединенными сглаживающими линиями без маркеров*», зададим значения *X*, имена рядам данных и нажимаем кнопку «*Готово*». В результате появится график табулированной функции *f*(*x*) на отрезке [1, 4] (рис. 4.17).

C4 fx =EXP(B4/2)-B4*B4*B4+10

	A	B	C	D
1				
2		**Предварительное табулирование**		
3		**x**	**y**	
4		1	10,6487	
5		1,2	10,0941	
6		1,4	9,26975	
7		1,6	8,12954	
8		1,8	6,6276	
9		2	4,71828	
10		2,2	2,35617	
11		2,4	-0,50388	
12		2,6	-3,9067	
13		2,8	-7,8968	
14		3	-12,5183	
15		3,2	-17,815	
16		3,4	-23,8301	
17		3,6	-30,6064	
18		3,8	-38,1861	
19		4	-46,6109	
20				

Рис. 4.16

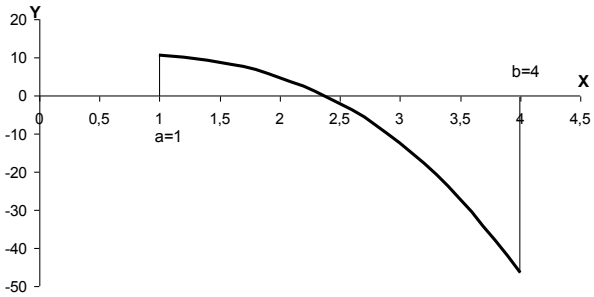


Рис. 4.17

Алгоритм нахождения корня

Для поиска корня уравнения используются итерационные вычисления – деление выбранного отрезка пополам.

Шаг 1. Первая итерация (результаты представлены на рис. 4.18):

– вводим номер итерации (i = 1) (B4);

– вводим значения границ отрезка (a_1 = 1 и b_1 = 4) (C4, D4);

– вычисляем значение координаты середины отрезка – $x_1 = (a_1 + b_1)/2$ (E4);

– вычисляем $f(a_1)$, $f(b_1)$, $f(x_1)$ (F4, G4, H4);

– вычисляем значение точности по формуле $|b_1 - a_1|/2$ (I4).

	A	B	C	D	E	F	G	H	I
1									
2		Итерация,	Границы		Приближение	Значения функции			Точность
3		i	a_i	b_i	к корню, x_i	$f(a_i)$	$f(b_i)$	$f(x_i)$	
4		1	1,000	4,000	2,500	10,649	-46,611	-2,135	1,500
5									

Рис. 4.18

Шаг 2. Вторая итерация (результаты представлены на рис. 4.21):

– вводим номер итерации (i = 2) (B5);

– определяем левую границу нового отрезка a_2 (C5) по алгоритму: если $f(a_1)\cdot f(x_1) < 0$, то $a_2 = a_1$, иначе $a_2 = x_1$ – для этого для ячейки C6 в строке формул вводится формула с использованием функции ЕСЛИ(), а именно, = ЕСЛИ(F4*H4<0;C4;E4) или с помощью диалогового окна (рис. 4.19);

– определяем правую границу нового отрезка b_2 (D5) по алгоритму: если $f(a_1)\cdot f(x_1) < 0$, то $b_2 = x_1$, иначе $b_2 = b_1$. Для этого аналогично предыдущему пункту для ячейки D5 в строке формул вводится формула с использованием функции ЕСЛИ(), а именно: =ЕСЛИ(F4*H4<0;E4;D4), или с помощью диалогового окна (рис. 4.20);

– вычисляем значения x_2, $f(a_2)$, $f(b_2)$, $f(x_2)$, точность второй итерации, для этого выделяем диапазон E4 : I4 и растягиваем его на один ряд вниз, в результате в ячейках E5 : I5 появятся значения второй итерации (рис. 4.21).

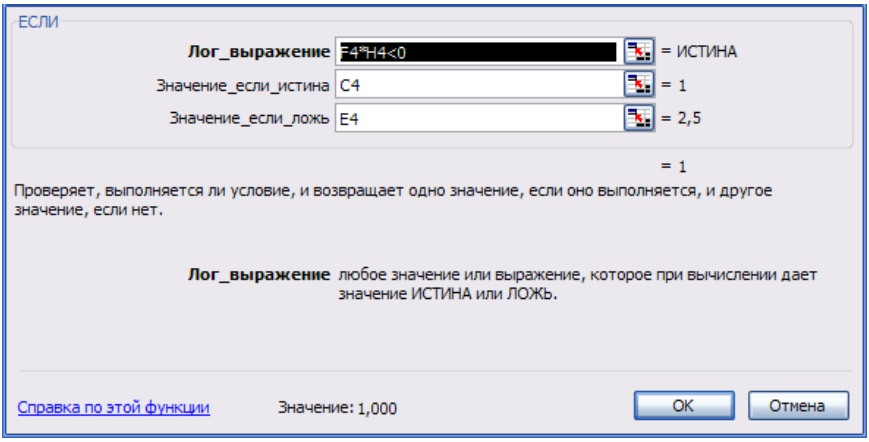


Рис. 4.19

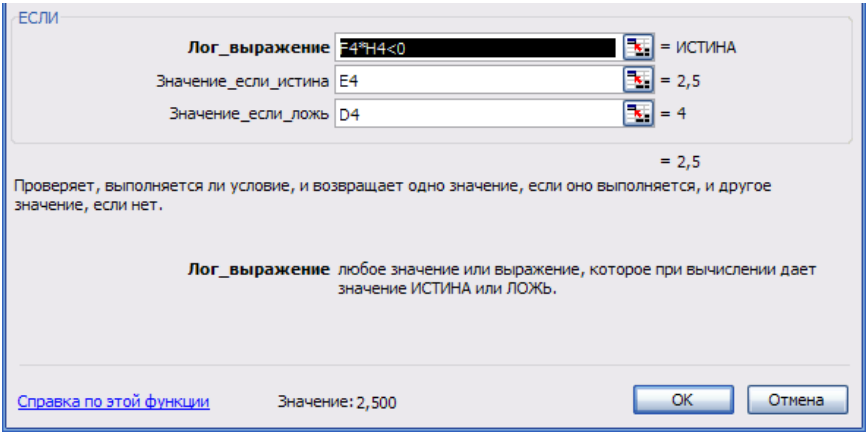


Рис. 4.20

	A	B	C	D	E	F	G	H	I
1									
2		Итерация,	Границы		Приближение	Значения функции			Точность
3		i	a_i	b_i	к корню, x_i	$f(a_i)$	$f(b_i)$	$f(x_i)$	
4		1	1,000	4,000	2,500	10,649	-46,611	-2,135	1,500
5		2	1,000	2,500	1,750	10,649	-2,135	7,040	0,750
6									

Рис. 4.21

Шаг 3. Итерации вычислений продолжаем до тех пор, пока не будет получена заданная точность (0,01). Для этого необходимо выделить диапазон B5 : I5 (рис. 4.22) и скопировать или растянуть его в нижние строки до тех пор, пока значение точности не станет меньше заданного значения (0,01) (рис. 4.23). Из рис. 4.23 видно, что на десятой итерации значение точности станет равным 0,006 < 0,01 и, следовательно, корень уравнения равен x = 2,365 (ячейка E12).

	A	B	C	D	E	F	G	H	I
1									
2		Итерация,	Границы		Приближение	Значения функции			Точность
3		i	a_i	b_i	к корню, x_i	$f(a_i)$	$f(b_i)$	$f(x_i)$	
4		1	1,000	4,000	2,500	10,649	-46,611	-2,135	1,500
5		2	1,000	2,500	1,750	10,649	-2,135	7,040	0,750
6									

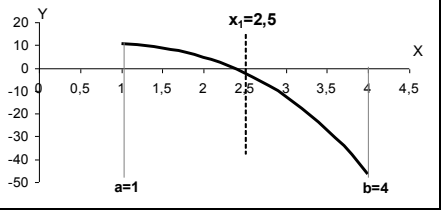
Рис. 4.22

	A	B	C	D	E	F	G	H	I
1									
2		Итерация, i	Границы		Приближение к корню, x_i	Значения функции			Точность
3			a_i	b_i		$f(a_i)$	$f(b_i)$	$f(x_i)$	
4		1	1,000	4,000	2,500	10,649	-46,611	-2,135	1,500
5		2	1,000	2,500	1,750	10,649	-2,135	7,040	0,750
6		3	1,750	2,500	2,125	7,040	-2,135	3,298	0,375
7		4	2,125	2,500	2,313	3,298	-2,135	0,812	0,188
8		5	2,313	2,500	2,406	0,812	-2,135	-0,602	0,094
9		6	2,313	2,406	2,359	0,812	-0,602	0,120	0,047
10		7	2,359	2,406	2,383	0,120	-0,602	-0,237	0,023
11		8	2,359	2,383	2,371	0,120	-0,237	-0,058	0,012
12		9	2,359	2,371	2,365	0,120	-0,058	0,031	0,006

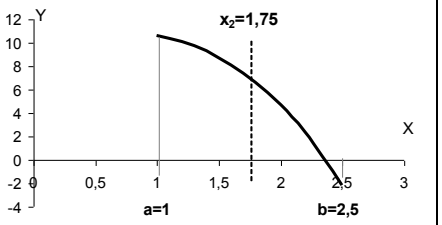
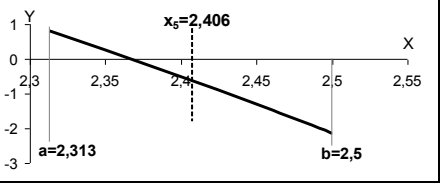
Рис. 4.23

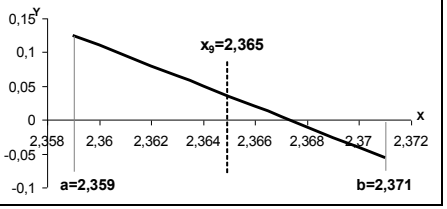
В табл. 4.1 приведена иллюстрация решения задачи.

Таблица 4.1

Итерация	Приближение к корню	График функции
1	2	3
Первая итерация	$x_1 = 2{,}5$	

Окончание табл. 4.1

1	2	3
Вторая итерация	$x_2 = 1{,}75$	
Пятая итерация	$x_5 = 2{,}406$	

Девятая итерация	$x_9 = 2{,}365$	

4.2.2. Задача решения нелинейного уравнения методом хорд

***Задача** 4.3.* Разработать алгоритм решения нелинейного уравнения $f(x) = e^{x/2} - x^3 + 10 = 0$ на отрезке [1, 4] методом хорд с точностью 0,01.

Алгоритм решения

Предварительное табулирование. Найдем значения функции $f(x) = e^{x/2} - x^3 + 10$ на отрезке [1, 4], используя шаг изменения переменной *x*, равный 0,2 (см. рис. 4.16). Построим график функции (см. рис. 4.17).

Алгоритм нахождения корня уравнения

Шаг 1. Первая итерация (результаты представлены на рис. 4.26):
– вводим номер итерации (i = 1) (B4);
– вводим значения границ отрезка (a_1 = 1 и b_1 = 4) (C4, D4);
– вычисляем значения функции на границах отрезка – $f(a_1)$, $f(b_1)$, (E4, F4) (рис. 4.24);

F4 =EXP(D4/2)-D4*D4*D4+10

	A	B	C	D	E	F	G	H	I
1									
2		Итерация, i	Границы отрезка		Значения функции на отрезке			Приближение к корню, x_i	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$	$f(x_i)$		
4		1	1,000	4,000	10,6487	-46,61			
5									

Рис. 4.24

– вычисляем приближение корня x_1 на отрезке $[a_1; b_1]$ по формуле $x_1 = \dfrac{b_1 \cdot f(a_1) - a_1 \cdot f(b_1)}{f(a_1) - f(b_1)}$ (H4) (рис. 4.25);

H4 =(D4*E4-C4*F4)/(E4-F4)

	A	B	C	D	E	F	G	H	I
1									
2		Итерация, i	Границы отрезка		Значения функции на отрезке			Приближение к корню, x_i	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$	$f(x_i)$		
4		1	1,000	4,000	10,6487	-46,61		1,558	

Рис. 4.25

– вычисляем значение функции в точке x_1 – $f(x_1)$ (G4) (рис. 4.26).

G4 =EXP(H4/2)-H4*H4*H4+10

	A	B	C	D	E	F	G	H	I
1									
2		Итерация, i	Границы отрезка		Значения функции на отрезке			Приближение к корню, x_i	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$	$f(x_i)$		
4		1	1,000	4,000	10,6487	-46,61	8,398	1,558	

Рис. 4.26

Шаг 2. Вторая итерация (результаты представлены на рис. 4.30):

– вводим номер итерации (i = 2) (B5);

– определяем левую границу нового отрезка a_2 (C5) по алгоритму: если $f(a_1) \cdot f(x_1) < 0$, то $a_2 = a_1$, иначе $a_2 = x_1$ – для этого для ячейки C5 в строке формул вводится формула с использованием функции ЕСЛИ(), а именно: =ЕСЛИ(C4*G4<0;C4;H4), или с помощью диалогового окна (рис. 4.27);

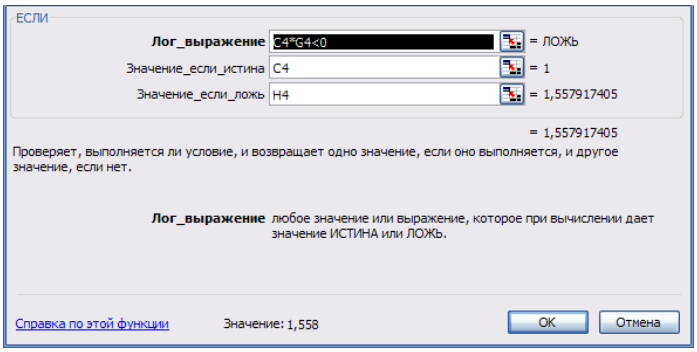


Рис. 4.27

– определяем правую границу нового отрезка b_2 (D5) по алгоритму: если $f(a_1)\cdot f(x_1) < 0$, то $b_2 = x_1$, иначе $b_2 = b_1$ – аналогично предыдущему пункту (рис. 4.28); значения границ отрезка на второй итерации приведены на рис. 4.29;

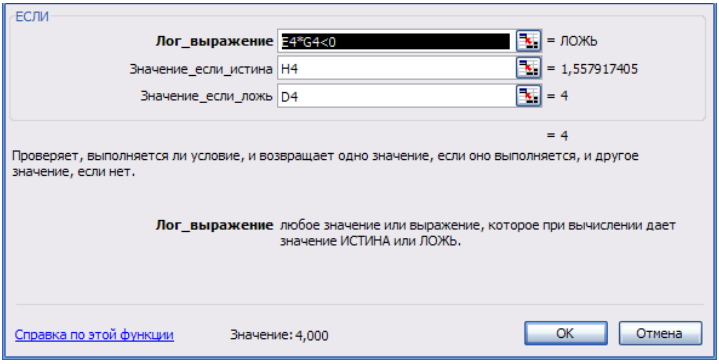


Рис. 4.28

D5 =ЕСЛИ(E4*G4<0;H4;D4)

	A	B	C	D	E	F	G	H	I
1									
2		Итерация, i	Границы отрезка		Значения функции на отрезке			Приближение к корню, x_i	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$	$f(x_i)$		
4		1	1,000	4,000	10,6487	-46,61	8,398	1,558	
5		2	1,558	4,000					

Рис. 4.29

– вычисляем значения функции на границах отрезка $f(a_2)$, $f(b_2)$ (E5, F5);

– вычисляем приближение корня x_2 на отрезке $[a_2; b_2]$ по формуле $x_2 = \dfrac{b_2 \cdot f(a_2) - a_2 \cdot f(b_2)}{f(a_2) - f(b_2)}$ (H5);

– вычисляем значения функции в точке x_2 – $f(x_2)$ (G5);

– вычисляем значение погрешности по формуле $|x_2 - x_1|$ (I5) (рис. 4.30).

I5 =ABS(H5-H4)

	A	B	C	D	E	F	G	H	I
1									
2		Итерация, i	Границы отрезка		Значения функции на отрезке			Приближение к корню, x_i	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$	$f(x_i)$		
4		1	1,000	4,000	10,6487	-46,61	8,398	1,558	
5		2	1,558	4,000	8,398	-46,611	5,428	1,931	0,373

Рис. 4.30

Шаг 3. Итерации вычислений продолжаем до тех пор, пока не будет получено значение точности, меньшее заданного. Для этого выделяем диапазон B5 : I5. Растягиваем его вниз до тех пор, пока значение точности (столбец I) не станет меньше заданного (0,01) (рис. 4.31).

	A	B	C	D	E	F	G	H	I
1									
2		Итерация, i	Границы отрезка		Значения функции на отрезке			Приближение к корню, x_i	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$	$f(x_i)$		
4		1	1,000	4,000	10,6487	-46,61	8,398	1,558	
5		2	1,558	4,000	8,398	-46,611	5,428	1,931	0,373
6		3	1,931	4,000	5,428	-46,611	3,034	2,147	0,216
7		4	2,147	4,000	3,034	-46,611	1,554	2,260	0,113
8		5	2,260	4,000	1,554	-46,611	0,761	2,316	0,056
9		6	2,316	4,000	0,761	-46,611	0,364	2,343	0,027
10		7	2,343	4,000	0,364	-46,611	0,172	2,356	0,013
11		8	2,356	4,000	0,172	-46,611	0,081	2,362	0,006

Рис. 4.31

Таким образом, из рисунка видно, что для поиска корня уравнения с заданной точностью достаточно восьми итераций и значение корня на заданном отрезке составляет 2,362 (значение x_8).

Иллюстрация решения приведена на рис. 4.32. Как видно из рисунка, проведенная хорда для каждой итерации стремится к искомому корню уравнения.

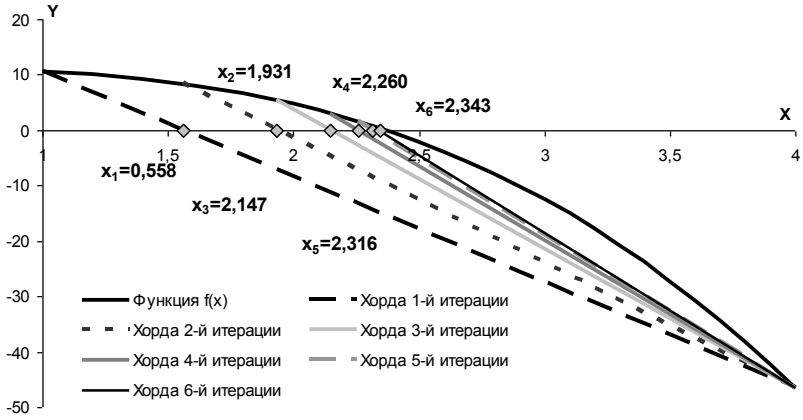


Рис. 4.32

4.2.3. Задача решения нелинейного уравнения методом касательных

Задача 4.4. Разработать алгоритм решения нелинейного уравнения $e^{x/2} - x^3 + 10 = 0$ на отрезке [1, 4] методом касательных с точностью 0,01. Положение корня в плоскости *XY* проиллюстрировать графически.

Алгоритм решения

Предварительное табулирование. Найдем значения функции $f(x) = e^{x/2} - x^3 + 10$ на отрезке [1, 4], используя шаг изменения переменной *x*, равный 0,2 (см. рис. 4.16). Построим график функции (см. рис. 4.17). Найдем первую и вторую производные функции *f*(*x*) по формулам

$$f'(x) = \frac{1}{2} e^{x/2} - 3x^2, \qquad (4.4)$$

$$f''(x) = \frac{1}{4} e^{x/2} - 6x. \qquad (4.5)$$

В дальнейшем при вычислении производных будем использовать полученные формулы.

Алгоритм нахождения корня уравнения

Шаг 1. Первая итерация (результаты представлены на рис. 4.35):

– вводим номер итерации (i = 1) (B4);

– вводим значения границ отрезка (a_1 = 1 и b_1 = 4) (C4, D4);

– вычисляем значения функции на границах отрезка $f(a_1)$, $f(b_1)$ (E4, F4);

– вычисляем значение второй производной в точке a_1: $f''(a_1)$ по формуле (4.5) (G4) (рис. 4.33);

G4 =0,25*EXP(C4/2) - 6*C4

	A	B	C	D	E	F	G	H	I	J
1										
2		Итерация, i	Границы отрезка		Значения функции на границах отрезка		$f''(a_i)$	Приближение к корню, x_i	$f(x_i)$	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$				
4		1	1,000	4,000	10,649	-46,611	-5,588			
5										

Рис. 4.33

– вычисляем приближение корня x_1 на отрезке $[a_1; b_1]$; для этого необходимо оценить выражение $f(a)\cdot f''(a_1)$: если $f(a)\cdot f''(a_1) > 0$, то касательная проводится к точке a и приближение x_1 вычисляется по формуле $x_1 = a - \dfrac{f(a)}{f'(a)}$; если $f(a)\cdot f''(a_1) \le 0$, то касательная проводится к точке b и приближение x_1 вычисляется по формуле $x_1 = b - \dfrac{f(b)}{f'(b)}$;

$f'(b)$, $f'(a)$ – это значения производной в точке b и a (G4), которые определяются по формуле (4.4) с использованием функции ЕСЛИ(), а именно: для ячейки H4 вводится формула =ЕСЛИ(E4*G4>0;C4-E4/(EXP(C4/2)/2 – 3*C4*C4);D4-F4/(EXP(D4/2)/2 – – 3*D4*D4)) или с помощью диалогового окна (рис. 4.34);

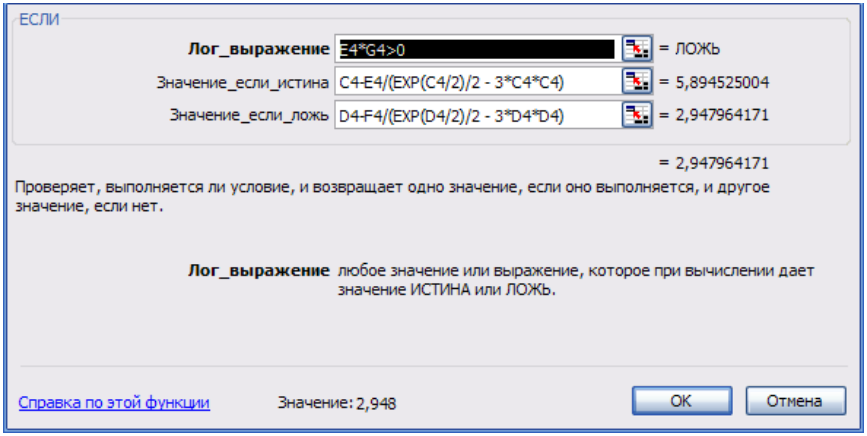


Рис. 4.34

– вычисляем значение функции в точке x_1: $f(x_1)$ (I4). (рис. 4.35).

I4 =EXP(H4/2)-H4*H4+10

	A	B	C	D	E	F	G	H	I	J
1										
2		Итерация, i	Границы отрезка		Значения функции на границах отрезка		$f''(a_i)$	Приближение к корню, x_i	$f(x_i)$	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$				
4		1	1,000	4,000	10,649	-46,611	-5,588	2,948	-11,253	

Рис. 4.35

Шаг 2. Вторая итерация (результаты представлены на рис. 4.40):

– вводим номер итерации (i = 2) (B5);

– определяем левую границу нового отрезка a_2 (C5): если $f(a_1)\cdot f(x_1) < 0$, то $a_2 = a_1$, иначе $a_2 = x_1$ – для этого для ячейки C5 в строке формул вводится формула с использованием функции ЕСЛИ(), а именно, =ЕСЛИ(E4*I4<0;C4;H4), или с помощью диалогового окна (рис. 4.36);

– определяем правую границу нового отрезка b_2 (D6): если $f(a_1)\cdot f(x_1) < 0$, то $b_2 = x_1$, иначе $b_2 = b_1$ – аналогично предыдущему пункту в строке формул вводится формула с использованием функции ЕСЛИ(), а именно, =ЕСЛИ(E4*I4<0;H4;D4), или с помощью диалогового окна (рис. 4.37); в результате получим значения, представленные на рис. 4.38;

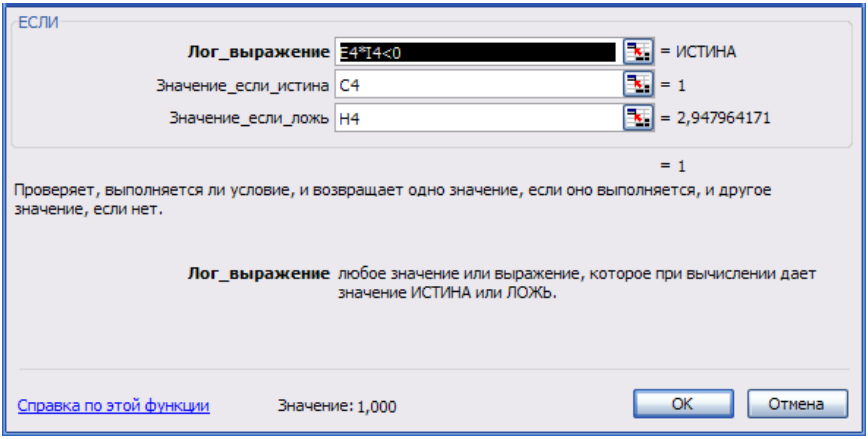


Рис. 4.36

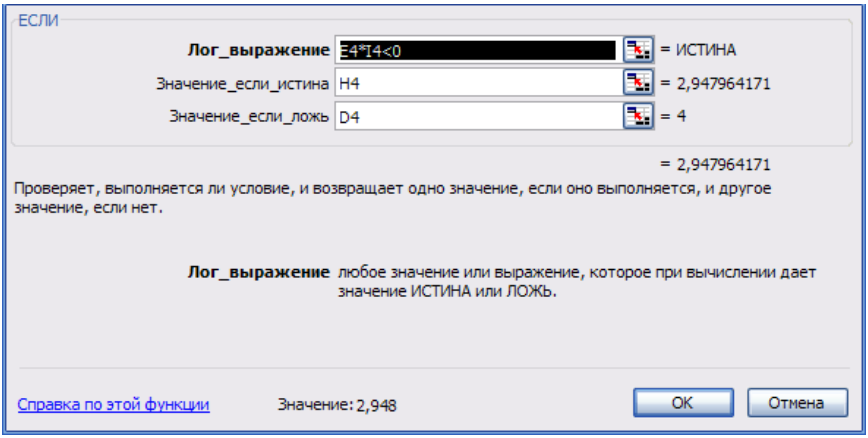


Рис. 4.37

D5 =ЕСЛИ(E4*I4<0;H4;D4)

	A	B	C	D	E	F	G	H	I	J
1										
2		Итерация, i	Границы отрезка		Значения функции на границах отрезка		$f''(a_i)$	Приближение к корню, x_i	$f(x_i)$	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$				
4		1	1,000	4,000	10,649	-46,611	-5,588	2,948	-11,253	
5		2	1,000	2,948						
6										

Рис. 4.38

– вычисляем значения функции на границах отрезка $f(a_2)$, $f(b_2)$ (E5, F5); значение второй производной в точке a_2: $f''(a_2)$ (G5), приближение корня x_2 на отрезке $[a_2; b_2]$ (H5); значение функции в точке x_2: $f(x_2)$ (I5) – для этого выделяем диапазон E4 : I4 и растягиваем маркер заполнения вниз на одну строку (рис. 4.39);

	A	B	C	D	E	F	G	H	I	J
1										
2		Итерация, i	Границы отрезка		Значения функции на границах отрезка		$f''(a_i)$	Приближение к корню, x_i	$f(x_i)$	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$				
4		1	1,000	4,000	10,649	-46,611	-5,588	2,948	-11,253	
5		2	1,000	2,948	10,649	-11,253	-5,588	2,477	-1,746	

Рис. 4.39

– вычисляем значение точности по формуле $|x_2 - x_1|$ (J5) (рис. 4.40).

J5 =ABS(H19-H4)

	A	B	C	D	E	F	G	H	I	J
1										
2		Итерация, i	Границы отрезка		Значения функции на границах отрезка		$f''(a_i)$	Приближение к корню, x_i	$f(x_i)$	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$				
4		1	1,000	4,000	10,649	-46,611	-5,588	2,948	-11,253	
5		2	1,000	2,948	10,649	-11,253	-5,588	2,477	-1,746	0,471

Рис. 4.40

Шаг 3. Итерации вычислений продолжаем до тех пор, пока не будет достигнута заданная точность. Для этого выделяем диапазон B5 : J5 и растягиваем его вниз до тех пор, пока значение точности (столбец J) не станет меньше заданной (0,01) (рис. 4.41).

	A	B	C	D	E	F	G	H	I	J
1										
2		Итерация, i	Границы отрезка		Значения функции на границах отрезка		$f''(a_i)$	Приближение к корню, x_i	$f(x_i)$	Точность
3			a_i	b_i	$f(a_i)$	$f(b_i)$				
4		1	1,000	4,000	10,649	-46,611	-5,588	2,948	-11,253	
5		2	1,000	2,948	10,649	-11,253	-5,588	2,477	-1,746	0,471
6		3	1,000	2,477	10,649	-1,746	-5,588	2,372	-0,076	0,105
7		4	1,000	2,372	10,649	-0,076	-5,588	2,367	-0,0002	0,005

Рис. 4.41

Таким образом, как видно из рис. 4.41, для поиска корня достаточно четырех итераций и значение $x = 2{,}367$.

Графическая иллюстрация решения представлена на рис. 4.42.

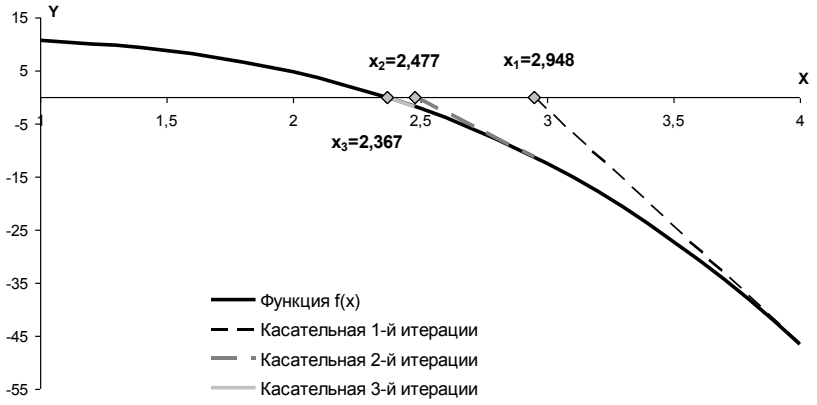


Рис. 4.42

Как видно из рисунка, проведение касательной на каждой итерации сужает рассматриваемый отрезок и каждое новое приближение корня приближается к искомому значению.

4.2.4. Задача решения нелинейного уравнения методом простых итераций

***Задача** 4.5.* Разработать алгоритм решения нелинейного уравнения $e^{x/2} - x^3 + 10 = 0$ на отрезке [1, 4] методом простых итераций с погрешностью 0,01.

Алгоритм решения

Представим исходное уравнение в следующем виде: $x = \sqrt[3]{e^{x/2} + 10}$. Поскольку выполняются следующие два условия:

1) функция $\varphi(x) = \sqrt[3]{e^{x/2} + 10}$ для всех $x \in [1, 4]$ удовлетворяет условию $\sqrt[3]{e^{1/2} + 10} \le \varphi(x) \le \sqrt[3]{e^{4/2} + 10}$, следовательно, $2{,}26 \le \varphi(x) \le 2{,}6$;

2) производная $\varphi'(x) = \dfrac{e^{x/2}}{6 \cdot \sqrt[3]{(10 + e^{x/2})^2}} < 1$,

то можно использовать метод простых итераций, по которому следующее приближения корня x_{i+1} вычисляется по формуле $x_{i+1} = \varphi(x_i)$.

Шаг 1. Первая итерация:

– введем номер итерации (i = 1) (B3);

– введем начальное приближение к корню уравнения (x_1 = 1) (C3) (рис. 4.43).

	A	B	C	D
1				
2		Итерация, i	Приближение к корню, x_i	Точность
3		1	1,000	

Рис. 4.43

Шаг 2. Вторая итерация:

– введем номер итерации (i = 2) (B4);

– вычислим следующее приближение к корню уравнения ($x_2 = \varphi(x_1) = \sqrt[3]{e^{x_1/2} + 10}$) (C4) (рис. 4.44);

– вычислим точность приближения по формуле $|x_2 - x_1|$ (D4) (рис. 4.45).

C4 =СТЕПЕНЬ(10+EXP(C3/2);1/3)

	A	B	C	D
1				
2		Итерация, i	Приближение к корню, x_i	Точность
3		1	1,000	
4		2	2,267	

Рис. 4.44

D4 =ABS(C4-C3)

	A	B	C	D
1				
2		Итерация, i	Приближение к корню, x_i	Точность
3		1	1,000	
4		2	2,267	1,267

Рис. 4.45

Шаг 3. Повторяем действия *шага 2*, пока не будет достигнута заданная точность. Для этого необходимо выделить диапазон ячеек B4 : D4 (рис. 4.46) и растянуть маркер заполнения вниз (рис. 4.47).

	A	B	C	D
1				
2		Итерация, i	Приближение к корню, x_i	Точность
3		1	1,000	
4		2	2,267	1,267

Рис. 4.46

	A	B	C	D
1				
2		Итерация, i	Приближение к корню, x_i	Точность
3		1	1,000	
4		2	2,267	1,267
5		3	2,358	0,091
6		4	2,366	0,009

Рис. 4.47

Таким образом, как видно из рис. 4.47, для поиска корня достаточно четырех итераций и значение корня $x = 2{,}366$.

4.2.5. Задача решения нелинейного уравнения с помощью надстройки «Поиск решения»

В этом разделе будут использованы возможности надстройки MS EXCEL *Поиск решения*. Они доступны с помощью команды *Сервис* → *Поиск решения*… Если этой команды в меню нет, ее необходимо установить. Для этого необходимо проделать следующее:

1. Выбрать пункт меню *Сервис* → *Надстройки*…
2. Установить в окне «Доступные надстройки» флажок для надстройки *Поиск решения*.
3. Далее возможны три ситуации:

– надстройка установлена в MS EXCEL, но не включена; в этом случае после нажатия кнопки *ОК* надстройка будет включена в MS EXCEL;

– надстройка не установлена в MS EXCEL, но находится в стандартных установочных каталогах Microsoft Office; в этом случае после нажатия кнопки *ОК* надстройка будет установлена и включена в состав MS EXCEL;

– надстройка не установлена в MS EXCEL, и ее нет в стандартных установочных каталогах Microsoft Office; в этом случае после нажатия кнопки *ОК* необходимо указать путь к установочному каталогу MS EXCEL на диске или другом носителе.

Можно также установить надстройку с помощью обращения к пункту *Установка и удаление программ* на панели управления Windows, для этого необходимо выбрать действие *Изменить* для

программы MS EXCEL или Microsoft Office и затем, следуя инструкциям в окне пошаговой установки, установить надстройку *Поиск решения*, указав для нее опцию *запускать с моего компьютера всегда*.

Алгоритм работы с надстройкой Поиск решения

В диалоговом окне Поиск решения есть три основных параметра:

1. «Установить целевую ячейку»;
2. «Изменяя ячейки»;
3. «Ограничения».

Сначала нужно заполнить поле *Установить целевую ячейку*. Целевая ячейка связана с другими ячейками этого рабочего листа с помощью формул, в ней происходит оптимизация результата. Можно выбрать поиск наименьшего или наибольшего значения для целевой ячейки или же установить конкретное значение.

Второй важный параметр — это параметр *Изменяя ячейки*. *Изменяемые ячейки* — это те ячейки, значения в которых будут изменяться для того, чтобы оптимизировать результат в целевой ячейке. Для поиска решения можно указать до 200 изменяемых ячеек. К изменяемым ячейкам предъявляются два основных требования. Они не должны содержать формул, и изменение их значений должно отражаться на изменении результата в целевой ячейке. Другими словами, целевая ячейка зависима от изменяемых ячеек.

Третий параметр, который нужно вводить для функции *Поиск решения*, – это ограничения. В качестве ограничений можно указать, например, диапазон изменения данных в изменяемых ячейках.

Задача 4.6. Разработать алгоритм решения нелинейного уравнения $f(x) = e^{x/2} - x^3 + 10 = 0$ на отрезке [–10, 4] с помощью надстройки MS EXCEL *Поиск решения*.

Алгоритм решения

Предварительное табулирование. Построим табулированную функцию $f(x) = e^{x/2} - x^3 + 10$, используя шаг изменения переменной x, равный 1 (рис. 4.48).

	A	B	C
1		Предварительное табулирование	
2		x	y
3		-10	-5.44017
4		-9	4.121308
5		-8	9.893918
6		-7	6.570778
7		-6	-2.79168
8		-5	-9.5825
9		-4	-7.54971
10		-3	1.460987
11		-2	9.22831
12		-1	8.782589
13		0	1
14		1	-5.69643
15		2	-1.70392
16		3	18.67434
17		4	62.16617

Рис. 4.48

Выделим диапазоны ячеек (значения *x* и *y*), на которых знак *y* меняется на противоположный, например, выделяем диапазон B3 : C4, поскольку при изменении *x* от –10 до –9, *y* изменяется от –5,44 до 4,12, т. е. *y* меняет знак (см. рис. 4.48).

Алгоритм нахождения корней уравнения

Шаг 1. Скопируем первые строки выделенных диапазонов, а именно: B3 :C 3 (*x* = –10, *y* = –5,44017), B6 : C6 (*x* = –7, *y* = 6,570778), B9 : C9 (*x* = –4, *y* = –7,54971), B13 : C13 (*x* = 0, *y* = 1), B15 : C15 (*x* = 2, *y* = –1,70392), отдельно (рис. 4.49).

Шаг 2. Запуск поиска корней уравнения с помощью функции *Поиск решения* для каждого выделенного диапазона ячеек (на *шаге 2*). Для этого необходимо выбрать пункт меню *Сервис – Поиск решения*... Появится диалоговое окно (рис. 4.50), в котором необходимо:

– в качестве целевой ячейки установить ячейку $F$3 – значение *y*;

– выбрать пункт с величиной, *равной значению 0*, т. е. значение *x* будет изменяться таким образом, чтобы значение *y* стало равным нулю;

– в поле *Изменяя ячейки* выбрать ячейку $E$3 – значение *x*;

– в поле списка *Ограничения* добавить ограничения на изменения ячейки $E$3 – значение *x* должно быть больше –10, но меньше –9.

E	F
Поиск корней уравнения	
x	y
-10	-5.44017
-7	6.570778
-4	-7.54971
0	1
2	-1.70392

Рис. 4.49

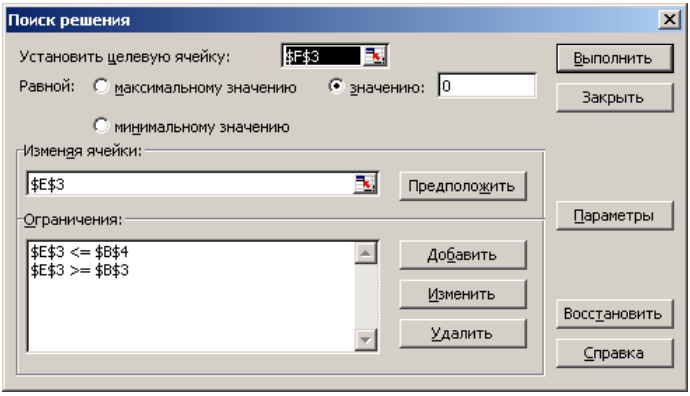


Рис. 4.50

После нажатия кнопки *Выполнить* будет выполнен поиск решения с заданными условиями. Затем после подтверждения в диалоговом окне *Результаты поиска решения* в ячейке E3 появится значение *x*, при котором значение *y* (ячейка F3) будет удовлетворять заданному условию (в нашем случае это 0) с заданной погрешностью. Погрешность можно задать, используя кнопку «Параметры» диалогового окна *Поиск решения*. После выполнения поиска решения для первого диапазона ячеек будут найдены следующие значения (диапазон результатов поиска E3 : F3) (рис. 4.51).

Шаг 3. Аналогично выполним пункт меню *Поиск решения* для каждого диапазона ячеек, на котором *y* меняет знак. Результаты поиска для всех диапазонов представлены на рис. 4.52. Таким обра-

зом, данное уравнение на заданном отрезке имеет пять корней: –9,42479; –6,283; –3,1459; 0,112097; 2,1345.

E	F
Поиск корней уравнения	
x	y
-9.42479	9.42E-07
-7	6.570778
-4	-7.54971
0	1
2	-1.70392

Рис. 4.51

E	F
Поиск корней уравнения	
x	y
-9.42479	9.42E-07
-6.283	8.52E-07
-3.1459	-4.7E-07
0.112097	-5.5E-07
2.1345	1.96E-07

Рис. 4.52

4.3. **Аппроксимация полиномами табулированных функций**

Задача *4.7.* По заданной таблице входных значений x, y (табл. 4.2) построить полином вида

$$y = a_0 + a_1x + a_2x^2 + a_3x^3$$

и определить его коэффициенты.

Таблица 4.2

x	0	1	2	3	4	5	6	7	8	9	10
y	3	8	20	78	163	348	503	962	1243	2064	2843
x	11	12	13	14	15	16	17	18	19	20	
y	3798	5100	6308	7899	9738	11843	14232	17200	19934	23283	
x	21	22	23	24	25	26	27	28	29	30	
y	26988	31067	35538	40419	45728	51483	57702	64403	71604	79323	

Построить графики по вычисленным значениям аппроксимационного полинома и сравнить их с табулированными значениями функции.

Алгоритм решения

Этап 1. Сформируем систему для нахождения аппроксимационного полинома:

$$\begin{cases} \sum\limits_{i=1}^{n}(a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3) = \sum\limits_{i=1}^{n} y_i, \\ \sum\limits_{i=1}^{n}(a_0 x_i + a_1 x_i^2 + a_2 x_i^3 + a_3 x_i^4) = \sum\limits_{i=1}^{n} x_i y_i, \\ \sum\limits_{i=1}^{n}(a_0 x_i^2 + a_1 x_i^3 + a_2 x_i^4 + a_3 x_i^5) = \sum\limits_{i=1}^{n} x_i^2 y_i, \\ \sum\limits_{i=1}^{n}(a_0 x_i^3 + a_1 x_i^4 + a_2 x_i^5 + a_3 x_i^6) = \sum\limits_{i=1}^{n} x_i^3 y_i, \end{cases} \quad (4.6)$$

где $n = 30$.

Этап 2. Преобразуем систему (4.6) к следующему виду:

$$\begin{cases} \sum\limits_{i=1}^{n} a_0 + a_1 \sum\limits_{i=1}^{n} x_i + a_2 \sum\limits_{i=1}^{n} x_i^2 + a_3 \sum\limits_{i=1}^{n} x_i^3 = \sum\limits_{i=1}^{n} y_i, \\ a_0 \sum\limits_{i=1}^{n} x_i + a_1 \sum\limits_{i=1}^{n} x_i^2 + a_2 \sum\limits_{i=1}^{n} x_i^3 + a_3 \sum\limits_{i=1}^{n} x_i^4) = \sum\limits_{i=1}^{n} x_i y_i, \\ a_0 \sum\limits_{i=1}^{n} x_i^2 + a_1 \sum\limits_{i=1}^{n} x_i^3 + a_2 \sum\limits_{i=1}^{n} x_i^4 + a_3 \sum\limits_{i=1}^{n} x_i^5 = \sum\limits_{i=1}^{n} x_i^2 y_i, \\ a_0 \sum\limits_{i=1}^{n} x_i^3 + a_1 \sum\limits_{i=1}^{n} x_i^4 + a_2 \sum\limits_{i=1}^{n} x_i^5 + a_3 \sum\limits_{i=1}^{n} x_i^6 = \sum\limits_{i=1}^{n} x_i^3 y_i. \end{cases} \quad (4.7)$$

Этап 3. Введем обозначения для вычисления сумм системы (4.7): $t_1 = \sum\limits_{i=1}^{n} x_i$, $t_2 = \sum\limits_{i=1}^{n} x_i^2$, $t_3 = \sum\limits_{i=1}^{n} x_i^3$, $t_4 = \sum\limits_{i=1}^{n} x_i^4$, $t_5 = \sum\limits_{i=1}^{n} x_i^5$, $t_6 = \sum\limits_{i=1}^{n} x_i^6$, $c_0 = \sum\limits_{i=1}^{n} y_i$, $c_1 = \sum\limits_{i=1}^{n} x_i y_i$, $c_2 = \sum\limits_{i=1}^{n} x_i^2 y_i$, $c_3 = \sum\limits_{i=1}^{n} x_i^3 y_i$ и сформируем новую систему:

$$\begin{cases} na_0 + a_1 t_1 + a_2 t_2 + a_3 t_3 = c_0, \\ a_0 t_1 + a_1 t_2 + a_2 t_3 + a_3 t_4 = c_1, \\ a_0 t_2 + a_1 t_3 + a_2 t_4 + a_3 t_5 = c_2, \\ a_0 t_3 + a_1 t_4 + a_2 t_5 + a_3 t_6 = c_3. \end{cases} \quad (4.8)$$

Этап 4. Вычислить значения введенных переменных t_1, t_2, …, t_6, c_0, c_1, c_2, c_3 для системы (4.8) в MS EXCEL можно двумя способами. Первый способ заключается в последовательном перемножении и суммировании значений переменных в ячейках. Результаты вычислений значений промежуточных переменных системы (4.8) приведены в табл. 4.3.

Таблица 4.3

$t_1 = \sum_{i=1}^{n} x_i$	$t_2 = \sum_{i=1}^{n} x_i^2$	$t_3 = \sum_{i=1}^{n} x_i^3$	$t_4 = \sum_{i=1}^{n} x_i^4$	$t_5 = \sum_{i=1}^{n} x_i^5$	$t_6 = \sum_{i=1}^{n} x_i^6$	$c_0 = \sum_{i=1}^{n} y_i$	$c_1 = \sum_{i=1}^{n} x_i y_i$	$c_2 = \sum_{i=1}^{n} x_i^2 y_i$	$c_3 = \sum_{i=1}^{n} x_i^3 y_i$
1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	3	0	0	0
1	1	1	1	1	1	8	8	8	8
2	4	8	16	32	64	20	40	80	160
3	9	27	81	243	729	78	234	702	2106
4	16	64	256	1024	4096	163	652	2608	10432
5	25	125	625	3125	15625	348	1740	8700	43500
6	36	216	1296	7776	46656	503	3018	18108	108648
7	49	343	2401	16807	117649	962	6734	47138	329966
8	64	512	4096	32768	262144	1243	9944	79552	636416
9	81	729	6561	59049	531441	2064	18576	167184	1504656
10	100	1000	10000	100000	1000000	2843	28430	284300	2843000
11	121	1331	14641	161051	1771561	3798	41778	459558	5055138
12	144	1728	20736	248832	2985984	5100	61200	734400	8812800
13	169	2197	28561	371293	4826809	6308	82004	1066052	13858676
14	196	2744	38416	537824	7529536	7899	110586	1548204	21674856
15	225	3375	50625	759375	11390625	9738	146070	2191050	32865750
16	256	4096	65536	1048576	16777216	11843	189488	3031808	48508928
17	289	4913	83521	1419857	24137569	14232	241944	4113048	69921816
18	324	5832	104976	1889568	34012224	17200	309600	5572800	100310400

Окончание табл. 4.3

1	2	3	4	5	6	7	8	9	10
19	361	6859	130321	2476099	47045881	19934	378746	7196174	136727306

20	400	8000	160000	3200000	64000000	23283	465660	9313200	186264000
21	441	9261	194481	4084101	85766121	26988	566748	11901708	249935868
22	484	10648	234256	5153632	113379904	31067	683474	15036428	330801416
23	529	12167	279841	6436343	148035889	35538	817374	18799602	432390846
24	576	13824	331776	7962624	191102976	40419	970056	23281344	558752256
25	625	15625	390625	9765625	244140625	45728	1143200	28580000	714500000
26	676	17576	456976	11881376	308915776	51483	1338558	34802508	904865208
27	729	19683	531441	14348907	387420489	57702	1557954	42064758	1135748466
28	784	21952	614656	17210368	481890304	64403	1803284	50491952	1413774656
29	841	24389	707281	20511149	594823321	71604	2076516	60218964	1746349956
30	900	27000	810000	24300000	729000000	79323	2379690	71390700	2141721000
Суммы									
465	9455	216225	5273999	13398745	350931215	631825	1543306	39240638	1025818234

Второй способ вычисления промежуточных переменных заключается в использовании встроенных математических функций СУММ(), СУММКВ() и СУММПРОИЗВ(). Распишем последовательность шагов вычисления переменных.

Шаг 1. На первом шаге вычислим параметр $t_1 = \sum_{i=1}^{n} x_i$, используя математическую функцию СУММ(A2 : A32), где A2 – адрес первой ячейки столбца со значениями x, A32 – адрес конечной ячейки столбца со значениями x.

Шаг 2. На втором шаге вычислим параметр $t_2 = \sum_{i=1}^{n} x_i^2$, используя математическую функцию СУММКВ(A2 : A32) (рис. 4.53, 4.54).

Шаг 3. На третьем шаге вычислим параметр $t_3 = \sum_{i=1}^{n} x_i^3$, используя математическую функцию СУММПРОИЗВ() (рис. 4.55, 4.56).

Шаг 4. На четвертом шаге вычислим параметр $t_4 = \sum_{i=1}^{n} x_i^4$, также используя математическую функцию СУММПРОИЗВ(), но в открывшемся диалоговом окне функции заполняем уже четыре поля координатами массива t_4 (рис. 4.57, 4.58).

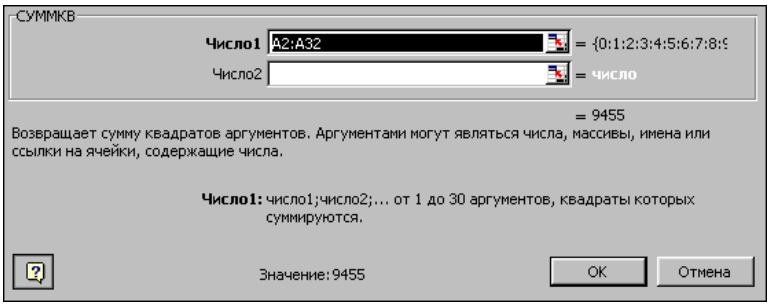


Рис. 4.53

D3	=	=СУММКВ(A2:A32)		
	A	B	C	D	E
1	x	y			
2	0	3	t_1=	465	
3	1	8	t_2=	9455	
4	2	20			
5	3	78			
6	4	163			

Рис. 4.54

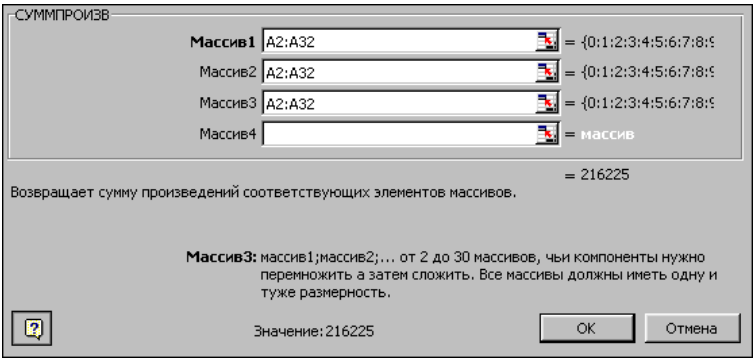


Рис. 4.55

D4 = =СУММПРОИЗВ(A2:A32;A2:A32;A2:A32)

	A	B	C	D	E	F
1	x	y				
2	0	3	t_1=	465		
3	1	8	t_2=	9455		
4	2	20	t_3=	216225		
5	3	78				
6	4	163				

Рис. 4.56

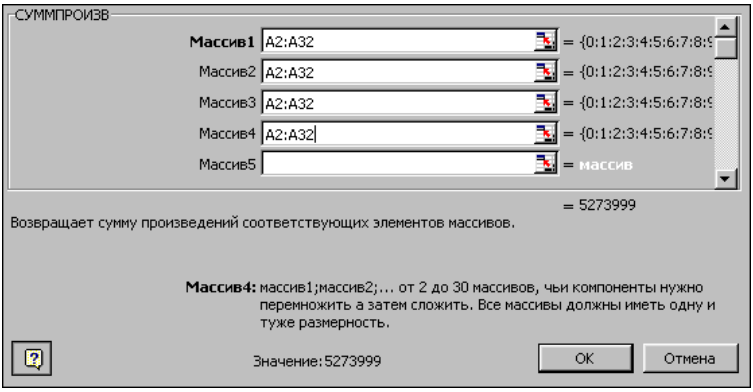


Рис. 4.57

D5 = =СУММПРОИЗВ(A2:A32;A2:A32;A2:A32;A2:A32)

	A	B	C	D	E	F	G
2	0	3	t_1=	465			
3	1	8	t_2=	9455			
4	2	20	t_3=	216225			
5	3	78	t_4=	5273999			
6	4	163					
7	5	348					

Рис. 4.58

Шаг 5. На пятом шаге вычислим параметр $t_5 = \sum_{i=1}^{n} x_i^5$, опять используя математическую функцию СУММПРОИЗВ(). Использование

этой функции обусловлено тем, что она позволяет перемножать и затем складывать указанные компоненты. Это же можно было делать, комбинируя различные функции, но, несмотря на кажущуюся «громоздкость» функции СУММПРОИЗВ, она оптимально подходит для требуемой операции, так как при внесении координат массивов удобно пользоваться технологией копирования (рис. 4.59, 4.60).

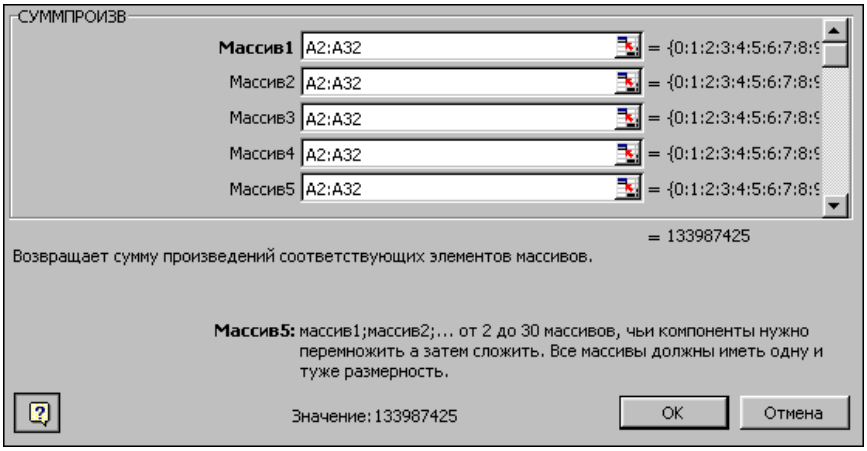


Рис. 4.59

D6 = =СУММПРОИЗВ(A2:A32;A2:A32;A2:A32;A2:A32;A2:A32)

	A	B	C	D	E	F	G	H
1	x	y						
2	0	3	t_1=	465				
3	1	8	t_2=	9455				
4	2	20	t_3=	216225				
5	3	78	t_4=	5273999				
6	4	163	t_5=	133987425				
7	5	348						
8	6	503						

Рис. 4.60

Шаг 6. На шестом шаге вычислим параметр $t_6 = \sum_{i=1}^{n} x_i^6$, используя математическую функцию СУММПРОИЗВ() (рис. 4.61, 4.62).

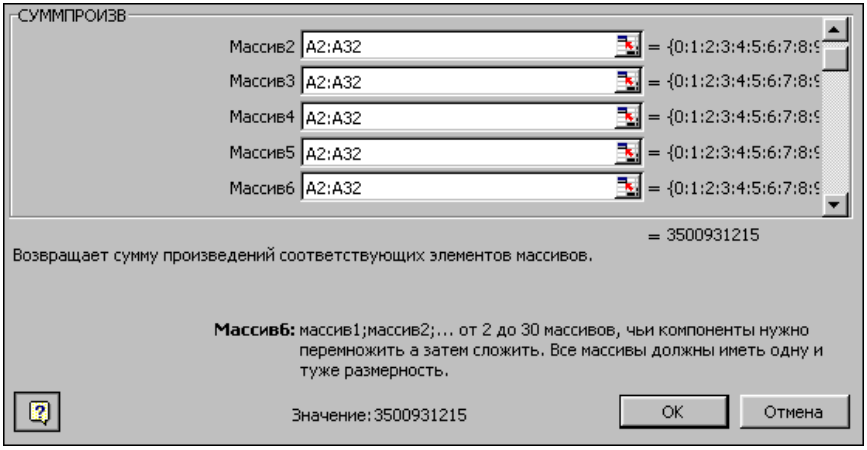


Рис. 4.61

D7 = =СУММПРОИЗВ(A2:A32;A2:A32;A2:A32;A2:A32;A2:A32;A2:A32)

	A	B	C	D	E	F	G	H
2	0	3	t_1=	465				
3	1	8	t_2=	9455				
4	2	20	t_3=	216225				
5	3	78	t_4=	5273999				
6	4	163	t_5=	133987425				
7	5	348	t_6=	3500931215				
8	6	503						
9	7	962						

Рис. 4.62

Шаг 7. На седьмом шаге вычислим параметр $c_0 = \sum_{i=1}^{n} y_i$, используя математическую функцию СУММ(B2 : B32), где B2 – адрес первой ячейки столбца со значениями y, B32 – адрес конечной ячейки столбца со значениями y.

Шаг 8. Используя математическую функцию СУММПРОИЗВ(), вычислим параметр $c_1 = \sum_{i=1}^{n} x_i y_i$ (рис. 4.63, 4.64).

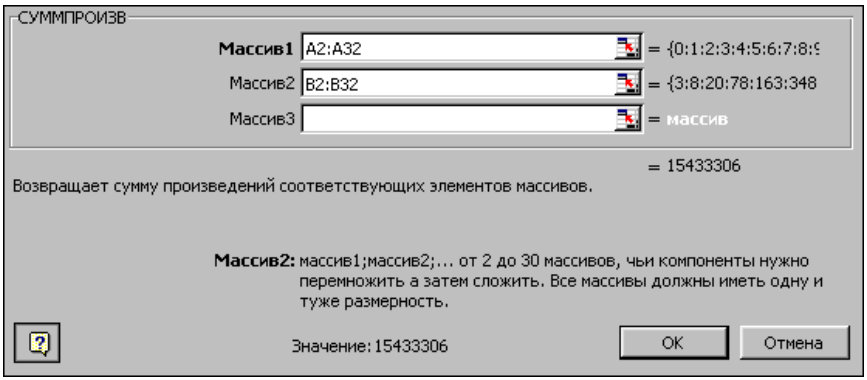


Рис. 4.63

D10 = =СУММПРОИЗВ(A2:A32;B2:B32)

	A	B	C	D	E	F
1	x	y				
2	0	3	t_1=	465		
3	1	8	t_2=	9455		
4	2	20	t_3=	216225		
5	3	78	t_4=	5273999		
6	4	163	t_5=	133987425		
7	5	348	t_6=	3500931215		
8	6	503				
9	7	962	c_0=	631825		
10	8	1243	c_1=	15433306		
11	9	2064				
12	10	2843				

Рис. 4.64

Шаг 9. Используя математическую функцию СУММПРОИЗВ(), вычислим параметр $c_2 = \sum_{i=1}^{n} x_i^2 y_i$ (рис. 4.65, 4.66).

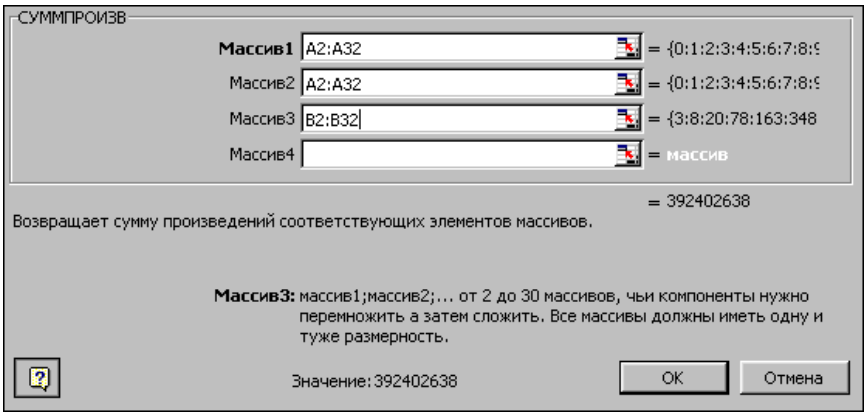


Рис. 4.65

D11 = =СУММПРОИЗВ(A2:A32;A2:A32;B2:B32)

	A	B	C	D	E	F
1	x	y				
2	0	3	t_1=	465		
3	1	8	t_2=	9455		
4	2	20	t_3=	216225		
5	3	78	t_4=	5273999		
6	4	163	t_5=	133987425		
7	5	348	t_6=	3500931215		
8	6	503				
9	7	962	c_0=	631825		
10	8	1243	c_1=	15433306		
11	9	2064	c_2=	392402638		
12	10	2843				
13	11	3798				

Рис. 4.66

Шаг 10. Используя математическую функцию СУММПРОИЗВ(), вычислим параметр $c_3 = \sum_{i=1}^{n} x_i^3 y_i$ (рис. 4.67, 4.68).

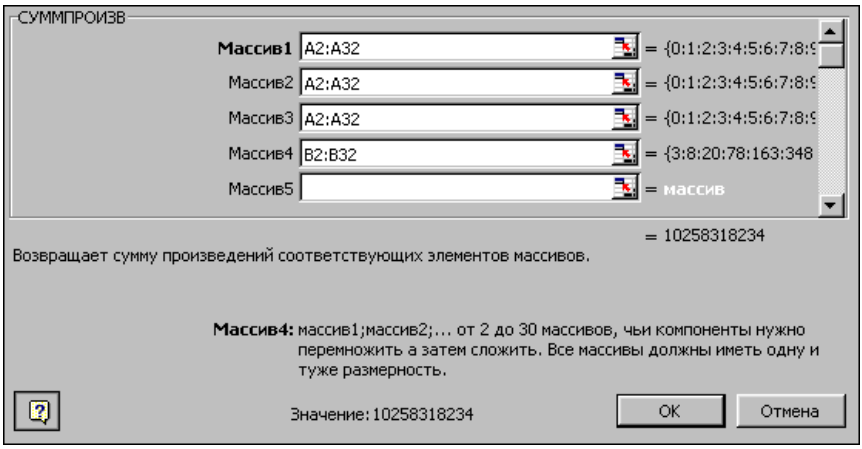


Рис. 4.67

D12 = =СУММПРОИЗВ(A2:A32;A2:A32;A2:A32;B2:B32)

	A	B	C	D	E	F	G
2	0	3	t_1=	465			
3	1	8	t_2=	9455			
4	2	20	t_3=	216225			
5	3	78	t_4=	5273999			
6	4	163	t_5=	133987425			
7	5	348	t_6=	3500931215			
8	6	503					
9	7	962	c_0=	631825			
10	8	1243	c_1=	15433306			
11	9	2064	c_2=	392402638			
12	10	2843	c_3=	10258318234			
13	11	3798					
14	12	5100					

Рис. 4.68

Этап 5. В результате проведенных вычислений и замены переменных t_1, t_2, …, t_6, c_0, c_1, c_2, c_3 в системе уравнений (4.8) на их значения, получаем систему уравнений

$$\begin{cases} 30a_0 + 465a_1 + 9455a_2 + 216225a_3 = 631825; \\ 465a_0 + 9455a_1 + 216225a_2 + 5273999a_3 = 15433306; \\ 9455a_0 + 216225a_1 + 5273999a_2 + 133987425a_3 = 392402638; \\ 216225a_0 + 5273999a_1 + 133987425a_2 + 3500931215a_3 = 10258318234. \end{cases} \quad (4.9)$$

Этап 6. Решим систему линейных уравнений (4.9) с помощью матричных операций в MS EXCEL.

Шаг 1. На первом шаге представим коэффициенты системы линейных уравнений (4.9) в виде матрицы коэффициентов T размером 4×4, а свободные члены линейных уравнений (4.7) – в виде матрицы-вектора C размером 4×1 (рис. 4.69). Значения a_0, a_1, a_2, a_3 будут представлены как вектор неизвестных $A_{4\times1}$. Таким образом, система уравнений (4.9) будет записана в матричном виде: $AT = C$. Чтобы найти вектор $A_{4\times1}$, необходимо умножить обе части полученного уравнения на матрицу $T^{-1}_{4\times4}$, обратную матрице $T_{4\times4}$: $ATT^{-1} = CT^{-1}$. В результате получим соотношение $A = CT^{-1}$.

	A	B	C	D	E	F	G	H
2	T (4x4)	30	465	9455	216225	C (1x4)	631825	
3		465	9455	216225	5273999		15433306	
4		9455	216225	5273999	133987425		392402638	
5		216225	5273999	133987425	3500931215		10258318234	
6								

Рис. 4.69

Шаг 2. На втором шаге найдем обратную матрицу $T^{-1}_{4\times4}$, используя функцию возвращения обратной матрицы МОБР(). Для этого выделим диапазон ячеек, в которых должна разместиться обратная матрица, и последовательно активизируем команды *Вставка функции → МОБР* (рис. 4.70). Зайдя в диалоговое окно МОБР(), заполним поле *Массив,* задав координаты матрицы $T_{4\times4}$ (рис. 4.71). Для завершения действия с функцией МОБР() нажимаем одновременно комбинацию клавиш *Ctrl + Shi ft + Enter* (рис. 4.72).

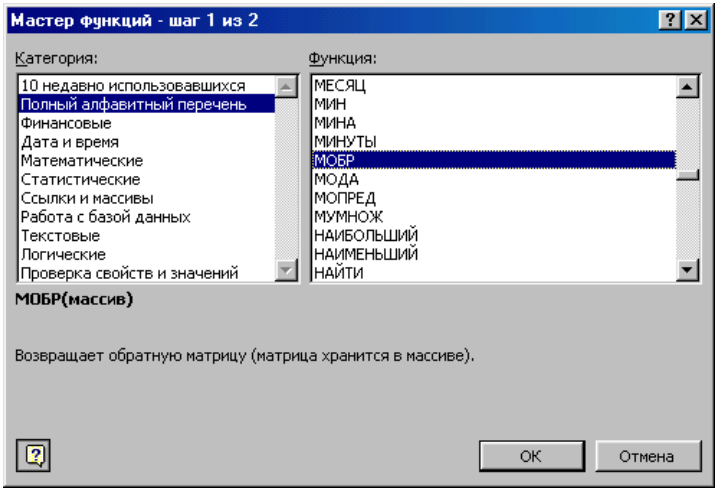


Рис. 4.70

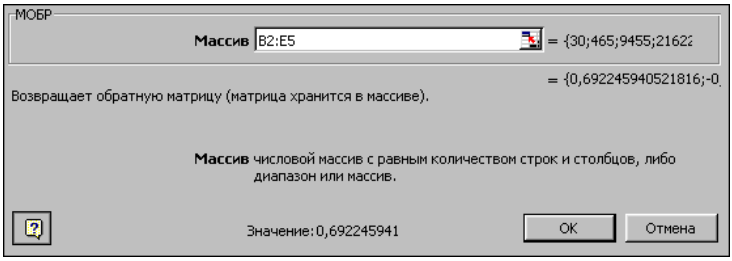


Рис. 4.71

B7 = {=МОБР(B2:E5)}

	A	B	C	D	E	F
2	T (4x4)	30	465	9455	216225	
3		465	9455	216225	5273999	
4		9455	216225	5273999	133987425	
5		216225	5273999	133987425	3500931215	
6						
7	T^{-1}(4x4)	0,692246	-0,169829	0,011129356	-0,00021286	
8		-0,169829	0,052275	-0,00377509	7,6219E-05	
9		0,011129	-0,003775	0,000288637	-6,0471E-06	
10		-0,000213	7,62E-05	-6,0471E-06	1,3004E-07	
11						

Рис. 4.72

Шаг 3. Найдем вектор неизвестных A, умножив матрицу T^{-1} на вектор C: $A = CT^{-1}$. Для этого выделим группу ячеек, в которых должен разместиться вектор неизвестных, и активизируем последовательно команды *Вставка функции* → *МУМНОЖ* (рис. 4.73).

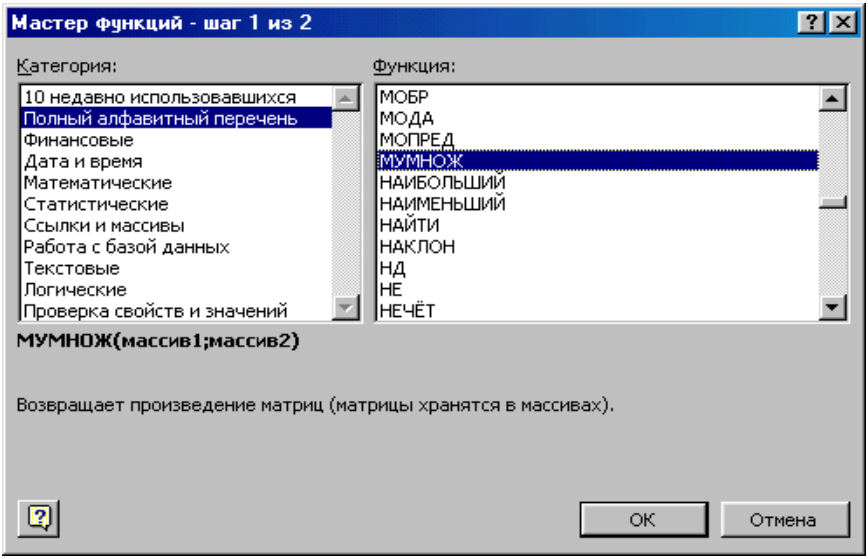


Рис. 4.73

Заполним поля *Массив1* и *Массив2*, задав координаты матрицы T^{-1} и вектора C (рис. 4.74).

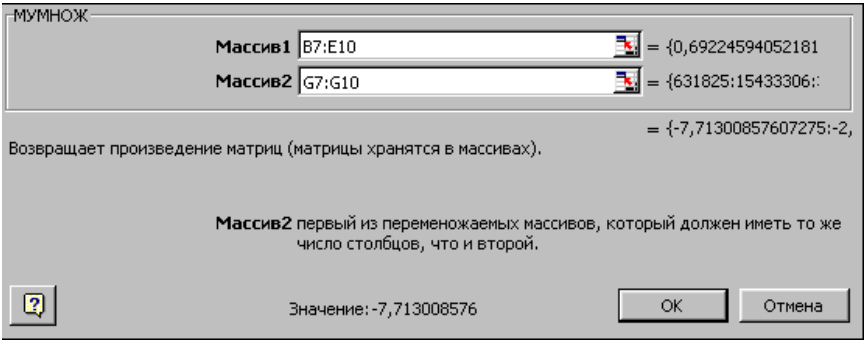


Рис. 4.74

Завершаем действие с функцией МУМНОЖ(), нажав одновременно комбинацию клавиш *Ctrl +Shift+Enter*. В результате на листе EXCEL получаем столбец со значениями элементов вектора неизвестных A размером 4×1 (рис. 4.75), т. е. при решении системы линейных уравнений (4.9) получен полином следующего вида:

$$y = a_0 + a_1x + a_2x^2 + a_3x^3,$$

где $a_0 = -7{,}71$, $a_1 = -2{,}39$, $a_2 = -1{,}09$, $a_3 = 2{,}98$.

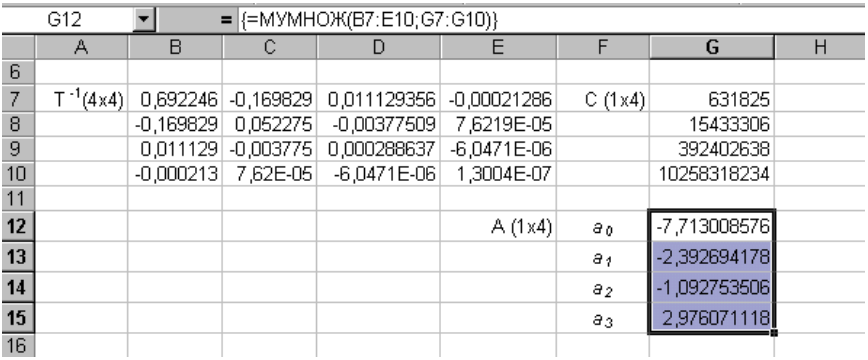


Рис. 4.75

Этап 7. Для построения функций y и $y_{\text{расч}}$, используем табличные (см. табл. 4.2) и расчетные данные. Расчетные данные величины $y_{\text{расч}}$ получаются путем прямой подстановки значений x в формулу полинома $y = -7{,}71 - 2{,}39x - 1{,}09x^2 + 2{,}98x^3$. Это можно сделать с помощью операторов MS EXCEL. Полученные значения $y_{\text{расч}}$ представлены в табл. 4.4.

Таблица 4.4

x	0	1	2	3	4	5	6	
$y_{\text{расч}}$	-7,71	-8,21	6,99	55,77	156,01	325,59	582,39	
x	7	8	9	10	11	12	13	14
$y_{\text{расч}}$	944,29	1429,17	2054,91	2839,39	3800,49	4956,09	6324,07	7922,31
x	15	16	17	18	19	20	21	22
$y_{\text{расч}}$	9768,69	11881,09	14277,39	16975,47	19993,21	23348,49	27059,19	31143,19
x	23	24	25	26	27	28	29	30
$y_{\text{расч}}$	35618,37	40502,61	45813,79	51569,79	57788,49	64487,77	71685,51	79399,59

Этап 8. По данным табл. 4.2 и 4.4 построим графики зависимости y от x (◇) и $y_{расч}$ от x (•) (рис. 4.76).

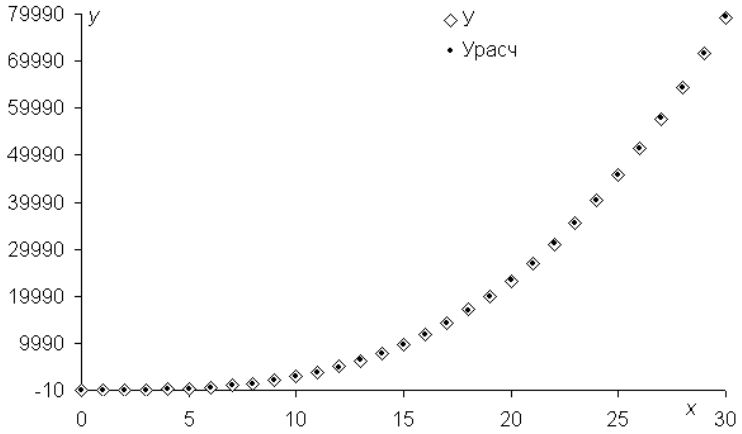


Рис. 4.76

На рис. 4.77 представлена часть диаграммы (см. рис. 4.76), на которой хорошо видно различие между заданными y и полученными $y_{расч}$ величинами. Тем не менее ход рядов y и $y_{расч}$ аналогичен (рис. 4.76), а значения y и $y_{расч}$ достаточно близки (см. табл. 4.2 и 4.4).

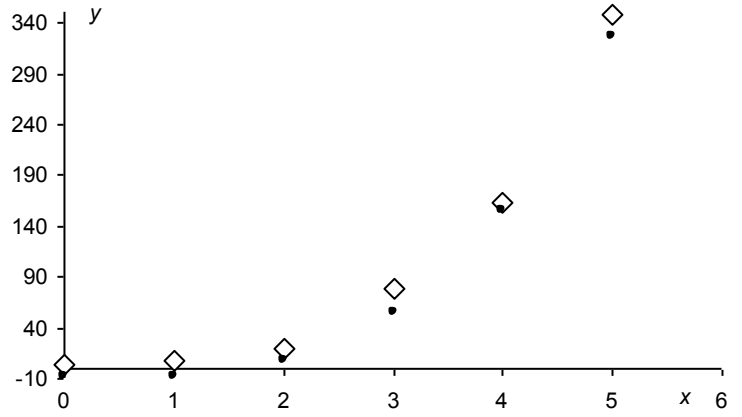


Рис. 4.77

4.4. **Численное дифференцирование и интегрирование функций**

Задача 4.8. Провести численное дифференцирование функции $y = \cos(2x)$, заданной таблично на участке $[-2\pi, 2\pi]$ с шагом 0,25, предварительно её протабулировав. Построить графики исходной и дифференциальной функций. При дифференцировании использовать формулы для левой, правой и центральной разностей (П 2.11)–(П 2.13).

Алгоритм решения

Шаг 1. Зададим начальное значение аргумента $x = -6{,}28$. Введем ряд значений X с шагом 0,25. Вычислим соответствующие значения Y. В результате проведенных действий будет получена таблица значений X и Y, представленная на рис. 4.78.

Шаг 2. Вычислим табличные значения производной по формуле центральных разностей (П 2.11) (рис. 4.79).

COS | =COS(2*A2)

A	B	C	D
X	y		
-6,28	S(2*A2)		
-6,03	0,874511		
-5,78	0,534931		
-5,53	0,064381		
-5,28	-0,42193		
-5,03	-0,80494		
-4,78	-0,99087		
-4,53	-0,9342		
-4,28	-0,64881		
-4,03	-0,20456		
⋮	⋮		
3,97	-0,08591		
4,22	-0,55305		
4,47	-0,88478		
4,72	-0,99988		
4,97	-0,87018		
5,22	-0,52743		
5,47	-0,05555		
5,72	0,429939		

Рис. 4.78

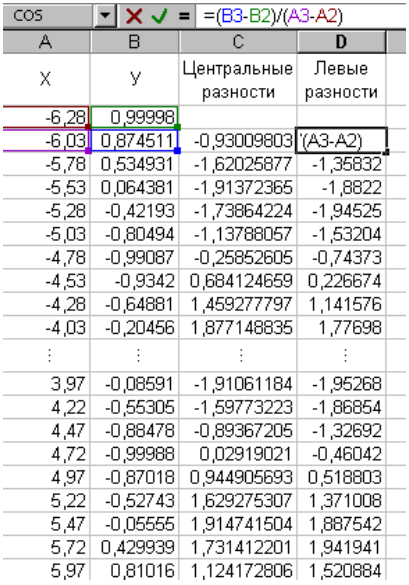
COS | =(B4-B2)/(A4-A2)

A	B	C	D
X	y	Центральные разности	
-6,28	0,99998		
-6,03	0,874511	i2)/(A4-A2)	
-5,78	0,534931	-1,62025877	
-5,53	0,064381	-1,91372365	
-5,28	-0,42193	-1,73864224	
-5,03	-0,80494	-1,13788057	
-4,78	-0,99087	-0,25852605	
-4,53	-0,9342	0,684124659	
-4,28	-0,64881	1,459277797	
-4,03	-0,20456	1,877148835	
⋮	⋮	⋮	
3,97	-0,08591	-1,91061184	
4,22	-0,55305	-1,59773223	
4,47	-0,88478	-0,89367205	
4,72	-0,99988	0,02919021	
4,97	-0,87018	0,944905693	
5,22	-0,52743	1,629275307	
5,47	-0,05555	1,914741504	

Рис. 4.79

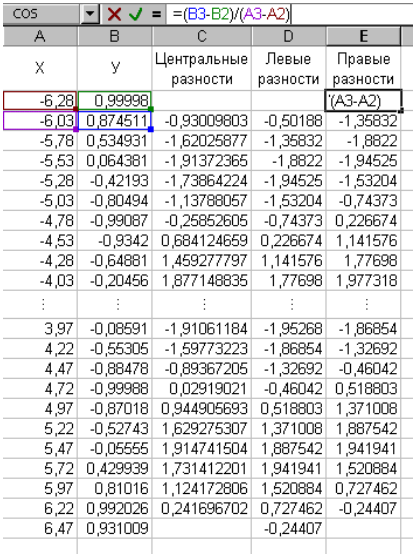
Шаг 3. Вычислим табличные значения производной по формуле левых разностей (П 2.12) (рис. 4.80).

Шаг 4. Вычислим табличные значения производной по формуле правых разностей (П 2.13) (рис. 4.81).



COS		=(B3-B2)/(A3-A2)	
A	B	C	D
X	y	Центральные разности	Левые разности
-6,28	0,99998		
-6,03	0,874511	-0,93009803	'(A3-A2)
-5,78	0,534931	-1,62025877	-1,35832
-5,53	0,064381	-1,91372365	-1,8822
-5,28	-0,42193	-1,73864224	-1,94525
-5,03	-0,80494	-1,13788057	-1,53204
-4,78	-0,99087	-0,25852605	-0,74373
-4,53	-0,9342	0,684124659	0,226674
-4,28	-0,64881	1,459277797	1,141576
-4,03	-0,20456	1,877148835	1,77698
⋮	⋮	⋮	⋮
3,97	-0,08591	-1,91061184	-1,95268
4,22	-0,55305	-1,59773223	-1,86854
4,47	-0,88478	-0,89367205	-1,32692
4,72	-0,99988	0,02919021	-0,46042
4,97	-0,87018	0,944905693	0,518803
5,22	-0,52743	1,629275307	1,371008
5,47	-0,05555	1,914741504	1,887542
5,72	0,429939	1,731412201	1,941941
5,97	0,81016	1,124172806	1,520884

Рис. 4.80



COS		=(B3-B2)/(A3-A2)		
A	B	C	D	E
X	y	Центральные разности	Левые разности	Правые разности
-6,28	0,99998			'(A3-A2)
-6,03	0,874511	-0,93009803	-0,50188	-1,35832
-5,78	0,534931	-1,62025877	-1,35832	-1,8822
-5,53	0,064381	-1,91372365	-1,8822	-1,94525
-5,28	-0,42193	-1,73864224	-1,94525	-1,53204
-5,03	-0,80494	-1,13788057	-1,53204	-0,74373
-4,78	-0,99087	-0,25852605	-0,74373	0,226674
-4,53	-0,9342	0,684124659	0,226674	1,141576
-4,28	-0,64881	1,459277797	1,141576	1,77698
-4,03	-0,20456	1,877148835	1,77698	1,977318
⋮	⋮	⋮	⋮	⋮
3,97	-0,08591	-1,91061184	-1,95268	-1,86854
4,22	-0,55305	-1,59773223	-1,86854	-1,32692
4,47	-0,88478	-0,89367205	-1,32692	-0,46042
4,72	-0,99988	0,02919021	-0,46042	0,518803
4,97	-0,87018	0,944905693	0,518803	1,371008
5,22	-0,52743	1,629275307	1,371008	1,887542
5,47	-0,05555	1,914741504	1,887542	1,941941
5,72	0,429939	1,731412201	1,941941	1,520884
5,97	0,81016	1,124172806	1,520884	0,727462
6,22	0,992026	0,241696702	0,727462	-0,24407
6,47	0,931009		-0,24407	

Рис. 4.81

Шаг 5. Используя функцию *Мастер диаграмм*, построим следующие графические зависимости:

$$y_1 = f(x), \quad y_2 = f_2(x), \quad y_3 = f_3(x), \quad y_4 = f_4(x),$$

где $f_2(x)$, $f_3(x)$, $f_4(x)$ – табличные функции, полученные на основе формул центральной, левой и правой разностей соответственно. Графики исходной (кривая *1*) и дифференциальных функций, полученных на основе формул для центральной (кривая *2*), левой (кривая *3*), правой (кривая *4*) разностей представлены на рис. 4.82.

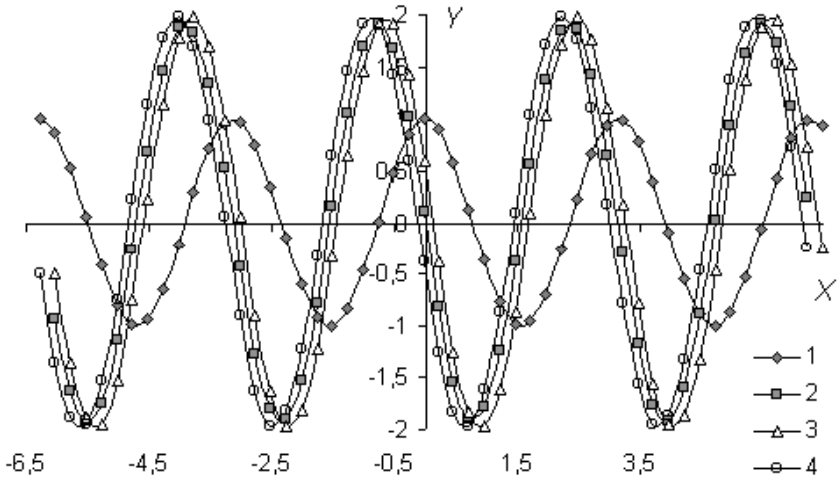


Рис. 4.82

***Задача** 4.9.* Изменив формулу функции $y = \cos 2x$ на $y = \cos 4x$, численно определить вторую, третью и четвертую производные, используя формулу центральных разностей. Построить графики функции $y = \cos 4x$ и вышеперечисленных производных. Определить значения сумм $S_1 = \sum_{i=1}^{n_1} f'(x_i)$, $S_2 = \sum_{i=1}^{n_2} f''(x_i) + \sum_{i=1}^{n_2} \cos(f''(x_i))$, $S_3 = \sum_{i=1}^{n_3} f'''(x_i) + \sum_{i=1}^{n_3} \cos(f'''(x_i))$, $S_4 = \sum_{i=1}^{n_4} f''''(x_i) + \sum_{i=1}^{n_4} \cos(f''''(x_i))$, где $f'(x_i)$, $f''(x_i)$, $f'''(x_i)$, $f''''(x_i)$ – производные первого, второго, третьего и четвертого порядка, вычисленные по формуле центральных разностей.

Алгоритм решения

Шаг 1. Используя значение аргумента *X* из задачи 4.8, вычислим значения *Y*. В результате проведенных действий будет получена таблица значений, представленная на рис. 4.83.

Шаг 2. Вычислим табличные значения производной *y′* по формуле центральных разностей (П 2.11) (рис. 4.84).

COS	=COS(4*A2)		
A	B	C	D
X	y		
-6,28	S(4*A2)		
-6,03	0,529537		
-5,78	-0,4277		
-5,53	-0,99171		
-5,28	-0,64395		
-5,03	0,295857		
-4,78	0,963652		
-4,53	0,74547		
-4,28	-0,15809		
-4,03	-0,91631		
⋮	⋮		
3,97	-0,98524		
4,22	-0,38828		
4,47	0,565666		
4,72	0,999537		
4,97	0,514438		
5,22	-0,44363		
5,47	-0,99383		
5,72	-0,6303		

Рис. 4.83

COS	=(B4-B2)/(A4-A2)		
A	B	C	D
X	y	y'	
-6,28	0,999919		
-6,03	0,529537	(A4-A2)	
-5,78	-0,4277	-3,0425	
-5,53	-0,99171	-0,4325	
-5,28	-0,64395	2,575134	
-5,03	0,295857	3,215201	
-4,78	0,963652	0,899228	
-4,53	0,74547	-2,24349	
-4,28	-0,15809	-3,32356	
-4,03	-0,91631	-1,34796	
⋮	⋮	⋮	
3,97	-0,98524	0,576204	
4,22	-0,38828	3,101808	
4,47	0,565666	2,775624	
4,72	0,999537	-0,10246	
4,97	0,514438	-2,88634	
5,22	-0,44363	-3,01653	
5,47	-0,99383	-0,37334	
5,72	-0,6303	2,613098	

Рис. 4.84

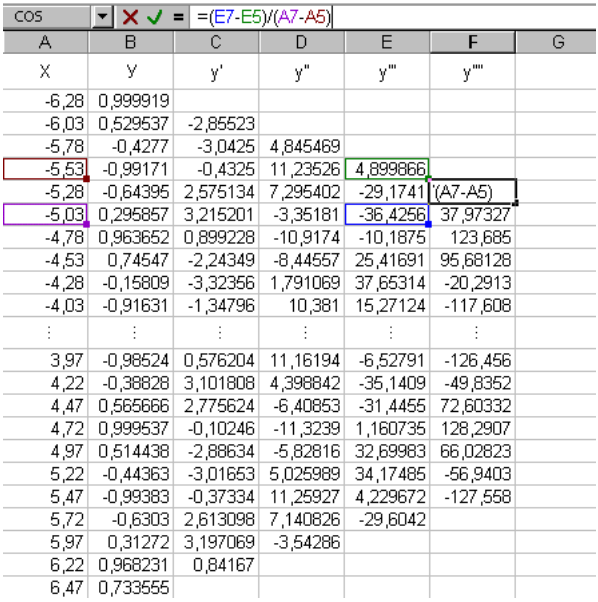
Шаг 3. Вычислим табличные значения производных y'' (рис. 4.85), y''' (рис. 4.86) и y'''' (рис. 4.87) по формуле центральных разностей (П 2.11).

COS	=(C5-C3)/(A5-A3)		
A	B	C	D
X	y	y'	y''
-6,28	0,999919		
-6,03	0,529537	-2,85523	
-5,78	-0,4277	-3,0425	(A5-A3)
-5,53	-0,99171	-0,4325	11,23526
-5,28	-0,64395	2,575134	7,295402
-5,03	0,295857	3,215201	-3,35181
-4,78	0,963652	0,899228	-10,9174
-4,53	0,74547	-2,24349	-8,44557
-4,28	-0,15809	-3,32356	1,791069
-4,03	-0,91631	-1,34796	10,381
⋮	⋮	⋮	⋮
3,97	-0,98524	0,576204	11,16194
4,22	-0,38828	3,101808	4,398842
4,47	0,565666	2,775624	-6,40853
4,72	0,999537	-0,10246	-11,3239
4,97	0,514438	-2,88634	-5,82816
5,22	-0,44363	-3,01653	5,025989
5,47	-0,99383	-0,37334	11,25927
5,72	-0,6303	2,613098	7,140826

Рис. 4.85

COS	=(D6-D4)/(A6-A4)			
A	B	C	D	E
X	y	y'	y''	y'''
-6,28	0,999919			
-6,03	0,529537	-2,85523		
-5,78	-0,4277	-3,0425	4,845469	
-5,53	-0,99171	-0,4325	11,23526	(A6-A4)
-5,28	-0,64395	2,575134	7,295402	-29,1741
-5,03	0,295857	3,215201	-3,35181	-36,4256
-4,78	0,963652	0,899228	-10,9174	-10,1875
-4,53	0,74547	-2,24349	-8,44557	25,41691
-4,28	-0,15809	-3,32356	1,791069	37,65314
-4,03	-0,91631	-1,34796	10,381	15,27124
⋮	⋮	⋮	⋮	⋮
3,97	-0,98524	0,576204	11,16194	-6,52791
4,22	-0,38828	3,101808	4,398842	-35,1409
4,47	0,565666	2,775624	-6,40853	-31,4455
4,72	0,999537	-0,10246	-11,3239	1,160735
4,97	0,514438	-2,88634	-5,82816	32,69983

Рис. 4.86



A	B	C	D	E	F	G
X	Y	y'	y''	y'''	y''''	
-6,28	0,999919					
-6,03	0,529537	-2,85523				
-5,78	-0,4277	-3,0425	4,845469			
-5,53	-0,99171	-0,4325	11,23526	4,899866		
-5,28	-0,64395	2,575134	7,295402	-29,1741	(A7-A5)	
-5,03	0,295857	3,215201	-3,35181	-36,4256	37,97327	
-4,78	0,963652	0,899228	-10,9174	-10,1875	123,685	
-4,53	0,74547	-2,24349	-8,44557	25,41691	95,68128	
-4,28	-0,15809	-3,32356	1,791069	37,65314	-20,2913	
-4,03	-0,91631	-1,34796	10,381	15,27124	-117,608	
⋮	⋮	⋮	⋮	⋮	⋮	
3,97	-0,98524	0,576204	11,16194	-6,52791	-126,456	
4,22	-0,38828	3,101808	4,398842	-35,1409	-49,8352	
4,47	0,565666	2,775624	-6,40853	-31,4455	72,60332	
4,72	0,999537	-0,10246	-11,3239	1,160735	128,2907	
4,97	0,514438	-2,88634	-5,82816	32,69983	66,02823	
5,22	-0,44363	-3,01653	5,025989	34,17485	-56,9403	
5,47	-0,99383	-0,37334	11,25927	4,229672	-127,558	
5,72	-0,6303	2,613098	7,140826	-29,6042		
5,97	0,31272	3,197069	-3,54286			
6,22	0,968231	0,84167				
6,47	0,733555					

Рис. 4.87

Шаг 4. Используя функцию *Мастер диаграмм,* построим следующие графические зависимости:

$$y_1 = f(x),$$

$$y_2 = y' = f_2(y),$$

$$y_3 = y'' = f_3(y'),$$

$$y_4 = y''' = f_4(y''),$$

$$y_5 = y'''' = f_5(y'''),$$

где $f_2(y)$, $f_3(y')$, $f_4(y'')$, $f_5(y''')$ – табличные функции, полученные на основе формулы центральной разности. Графики исходной (кривая *1*) и дифференциальных функций первого (кривая *2*), второго (кривая *3*), третьего (кривая *4*) и четвертого (кривая *5*) порядков, полученных на основе формулы для центральной разностей, представлены на рис. 4.88.

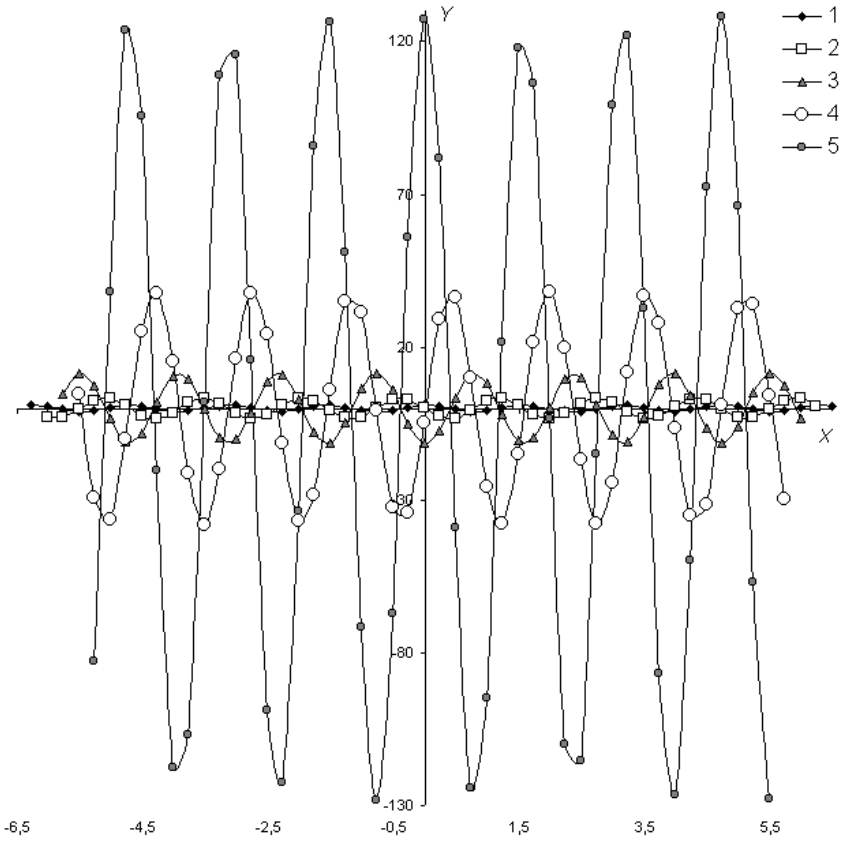


Рис. 4.88

Шаг 5. Найдем значения сумм

$$S_1 = \sum_{i=1}^{n_1} f'(x_i),$$

$$S_2 = \sum_{i=1}^{n_2} f''(x_i) + \sum_{i=1}^{n_2} \cos(f''(x_i)),$$

$$S_3 = \sum_{i=1}^{n_3} f'''(x_i) + \sum_{i=1}^{n_3} \cos(f'''(x_i)),$$

$$S_4 = \sum_{i=1}^{n_4} f''''(x_i) + \sum_{i=1}^{n_4} \cos(f''''(x_i))$$

с помощью встроенных функций EXCEL (рис. 4.89). В результате получим $S_1 = 0{,}345$; $S_2 = 15{,}204$; $S_3 = -19{,}734$; $S_4 = -7{,}404$.

COS | =COS(F6)

A	B	C	D	E	F	G	H	I
X	y	y'	y''	y'''	y''''	cos(y'')	cos(y''')	cos(y'''')
-6,28	0,999919							
-6,03	0,529537	-2,85523						
-5,78	-0,4277	-3,0425	4,845469			0,132687		
-5,53	-0,99171	-0,4325	11,23526	4,899866		0,237395	0,186380985	
-5,28	-0,64395	2,575134	7,295402	-29,1741	-82,6509	0,529982	-0,6217624	=COS(F6)
-5,03	0,295857	3,215201	-3,35181	-36,4256	37,97327	-0,97799	0,292901669	0,962654
-4,78	0,963652	0,899228	-10,9174	-10,1875	123,685	-0,07811	-0,72295403	-0,39672
-4,53	0,74547	-2,24349	-8,44557	25,41691	95,68128	-0,55768	0,959895045	0,136862
-4,28	-0,15809	-3,32356	1,791069	37,65314	-20,2913	-0,2185	0,998943293	0,12866
-4,03	-0,91631	-1,34796	10,381	15,27124	-117,608	-0,57661	-0,9061435	-0,20017
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3,97	-0,98524	0,576204	11,16194	-6,52791	-126,456	0,165594	0,97020391	0,702561
4,22	-0,38828	3,101808	4,398842	-35,1409	-49,8352	-0,30844	-0,83459234	0,908867
4,47	0,565666	2,775624	-6,40853	-31,4455	72,60332	0,992155	0,999561761	-0,9405
4,72	0,999537	-0,10246	-11,3239	1,160735	128,2907	0,322482	0,398665391	-0,8705
4,97	0,514438	-2,88634	-5,82816	32,69983	66,02823	0,898249	0,282974594	-0,9985
5,22	-0,44363	-3,01653	5,025989	34,17485	-56,9403	0,308486	-0,9276715	0,924282
5,47	-0,99383	-0,37334	11,25927	4,229672	-127,558	0,260647	-0,46418709	-0,31807
5,72	-0,6303	2,613098	7,140826	-29,6042		0,654224	-0,23856029	
5,97	0,31272	3,197069	-3,54286			-0,92057		
6,22	0,968231	0,84167						
6,47	0,733555							

Рис. 4.89

Задача 4.10. Используя метод трапеций, вычислить значение интеграла (шаг интегрирования равен 0,01)

$$S = \int_1^2 x^2 dx.$$

Алгоритм решения

Шаг 1. Распишем квадратурную сумму для вычисляемого интеграла в соответствии с формулой суммирования интеграла по методу трапеций:

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{2}\left[f(x_0) + 2f(x_0+h) + \ldots + 2f(x_0+h(n-1)) + f(x_n)\right] \quad (4.11)$$

$$S = \int_1^2 x^2 dx = \frac{0{,}01}{2}\left[1 + 2\cdot(1+0{,}01)^2 + \ldots + 2\cdot(1+0{,}01\cdot 99)^2 + 2^2\right].$$

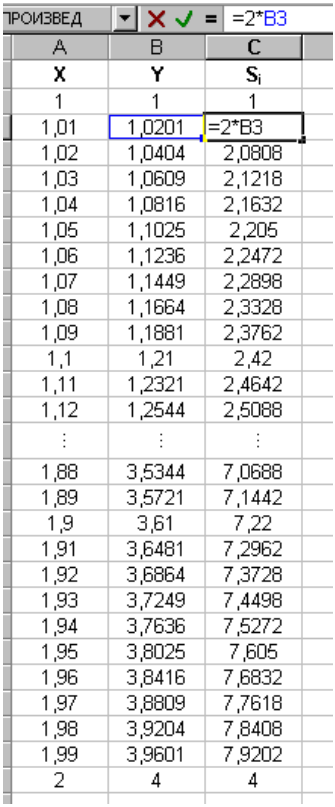
Шаг 2. Построим табулированную подынтегральную функцию $y = x^2$, используя шаг 0,01. В соответствии с формулой трапеций (4.11) на концах табулированной функции принять $f(x_0) = 1$, $f(x_n) = 2$. Значения $f(x_0+h)$, $f(x_0+2h)$, …, $f(x_0+(n-1)h)$ требуется умножить на 2. В результате получим столбец S_i (рис. 4.90).

Шаг 3. Вычислим $S_1 = \sum_{i=1}^{n} S_i$ и $S = \frac{0{,}01}{2} S_1$, где S_1 – численное значение интеграла (рис. 4.90).

Шаг 4. С целью проверки полученного численного значения интеграла ($S = 2{,}33$) определим его аналитическое значение:

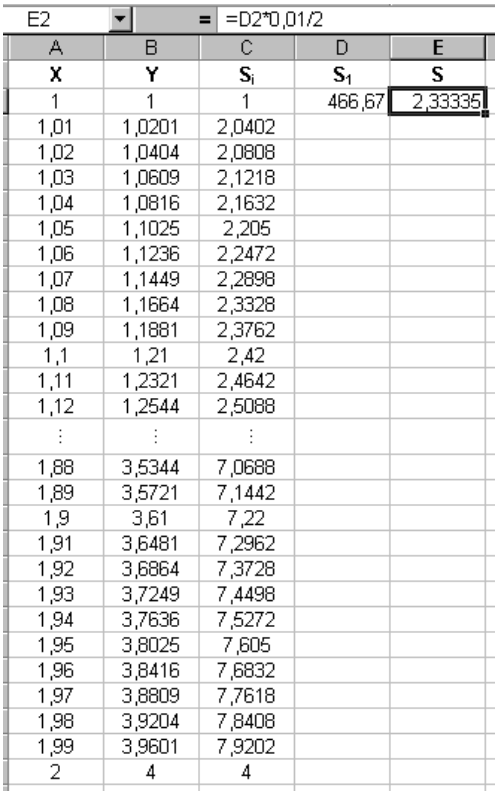
$$S = \int_1^2 x^2 dx = \frac{x^3}{3}\Bigg|_1^2 = \frac{2^3}{3} - \frac{1^3}{3} = \frac{7}{3} = 2{,}33.$$

Численное значение интеграла ($S = 2{,}33$) и его аналитическое значение совпали.



ПРОИЗВЕД		=2*B3
A	B	C
X	Y	S_i
1	1	1
1,01	1,0201	=2*B3
1,02	1,0404	2,0808
1,03	1,0609	2,1218
1,04	1,0816	2,1632
1,05	1,1025	2,205
1,06	1,1236	2,2472
1,07	1,1449	2,2898
1,08	1,1664	2,3328
1,09	1,1881	2,3762
1,1	1,21	2,42
1,11	1,2321	2,4642
1,12	1,2544	2,5088
⋮	⋮	⋮
1,88	3,5344	7,0688
1,89	3,5721	7,1442
1,9	3,61	7,22
1,91	3,6481	7,2962
1,92	3,6864	7,3728
1,93	3,7249	7,4498
1,94	3,7636	7,5272
1,95	3,8025	7,605
1,96	3,8416	7,6832
1,97	3,8809	7,7618
1,98	3,9204	7,8408
1,99	3,9601	7,9202
2	4	4

Рис. 4.90



E2			=D2*0,01/2	
A	B	C	D	E
X	Y	S_i	S_1	S
1	1	1	466,67	2,33335
1,01	1,0201	2,0402		
1,02	1,0404	2,0808		
1,03	1,0609	2,1218		
1,04	1,0816	2,1632		
1,05	1,1025	2,205		
1,06	1,1236	2,2472		
1,07	1,1449	2,2898		
1,08	1,1664	2,3328		
1,09	1,1881	2,3762		
1,1	1,21	2,42		
1,11	1,2321	2,4642		
1,12	1,2544	2,5088		
⋮	⋮	⋮		
1,88	3,5344	7,0688		
1,89	3,5721	7,1442		
1,9	3,61	7,22		
1,91	3,6481	7,2962		
1,92	3,6864	7,3728		
1,93	3,7249	7,4498		
1,94	3,7636	7,5272		
1,95	3,8025	7,605		
1,96	3,8416	7,6832		
1,97	3,8809	7,7618		
1,98	3,9204	7,8408		
1,99	3,9601	7,9202		
2	4	4		

Рис. 4.91

4.5. **Решение дифференциальных уравнений**

4.5.1. Решение дифференциальных уравнений методом Рунге-Кутта

Задача *4.11.* Решить дифференциальное уравнение $y' = 2y + e^x$ на отрезке [0, 2] методом Рунге-Кутта четвертого порядка. Начальное условие Коши $y(0) = 1$.

Алгоритм решения

Шаг 1. Распишем формулы для вычисления коэффициентов и значения уравнения.

Для *i*-й итерации коэффициенты k_1, k_2, k_3, k_4 вычисляются с использованием значений x_i, y_i:

$$k_1 = f(x_i, y_i) = 2y_i + e^{x_i};$$

$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h \cdot k_1}{2}\right) = 2 \cdot \left(y_i + \frac{h \cdot k_1}{2}\right) + e^{x_i + \frac{h}{2}};$$

$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h \cdot k_2}{2}\right) = 2 \cdot \left(y_i + \frac{h \cdot k_2}{2}\right) + e^{x_i + \frac{h}{2}};$$

$$k_4 = f(x_i + h, y_i + h \cdot k_3) = 2 \cdot (y_i + h \cdot k_3) + e^{x_i + h}.$$

Для $i + 1$ итерации значение $y_{i+1} = y_i + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$

Шаг изменения переменной x ($x_{i+1} - x_i$) установить равным 0,1 (ячейка D1).

Шаг 2. Первая итерация (результаты представлены на рис. 4.97).

Используя начальные значения $x_1 = 0$ и $y_1 = 1$, вычислить значения коэффициентов k_1, k_2, k_3, k_4 и значение y_2 по формулам, приведенным выше. Для этого необходимо сделать следующее (рис. 4.92–4.97):

– ввести значение шага для переменной x в ячейку D1, ссылку на которую в формулах следует сделать абсолютной (обозначить ее $D$1), таким образом при копировании формул ссылка на эту ячейку не сместится;

– ввести начальные значения x, y (B4, B9) (рис. 4.92);

– вычислить коэффициент k_1 (B5) (рис. 4.93);

– вычислить коэффициент k_2 (B6) (рис. 4.94);

– вычислить коэффициент k_3 (B7) (рис. 4.95);

– вычислить коэффициент k_4 (B8) (рис. 4.96);

– вычислить значение y для следующей итерации (C9) (рис. 4.97).

	A	B	C	D
1	**Шаг изменения x**			0,1
2				
3	**Итерация**	**1**		
4	**x**	0,00		
5	k_1			
6	k_2			
7	k_3			
8	k_4			
9	**y**	1,00		
10				

Рис. 4.92

B5 =2*B9+EXP(B4)

	A	B	C	D
1	**Шаг изменения x**			0,1
2				
3	**Итерация**	**1**		
4	**x**	0,00		
5	k_1	3,00		
6	k_2			
7	k_3			
8	k_4			
9	**y**	1,00		
10				

Рис. 4.93

B6 =2*(B9+$D$1*B5/2)+EXP(B4+$D$1/2)

	A	B	C	D	E	F	G
1	**Шаг изменения x**			0,1			
2							
3	**Итерация**	**1**					
4	**x**	0,00					
5	k_1	3,00					
6	k_2	3,35					
7	k_3						
8	k_4						
9	**y**	1,00					
10							

Рис. 4.94

B7 =2*(B9+$D$1*B6/2)+EXP(B4+$D$1/2)

	A	B	C	D	E	F	G
1	**Шаг изменения x**			0,1			
2							
3	**Итерация**	**1**					
4	**x**	0,00					
5	k_1	3,00					
6	k_2	3,35					
7	k_3	3,39					
8	k_4						
9	**y**	1,00					
10							

Рис. 4.95

B8 =2*(B9+$D$1*B7)+EXP(B4+$D$1)

	A	B	C	D	E	F
1	**Шаг изменения x**			0,1		
2						
3	**Итерация**	**1**				
4	**x**	0,00				
5	k_1	3,00				
6	k_2	3,35				
7	k_3	3,39				
8	k_4	3,78				
9	**y**	1,00				
10						

Рис. 4.96

C9 =B9+$D$1*(B5+2*B6+2*B7+B8)/6

	A	B	C	D	E	F
1	**Шаг изменения x**			0,1		
2						
3	**Итерация**	**1**	**2**			
4	**x**	0,00				
5	k_1	3,00				
6	k_2	3,35				
7	k_3	3,39				
8	k_4	3,78				
9	**y**	1,00	1,34			
10						

Рис. 4.97

Шаг 3. Вторая итерация (результаты представлены на рис. 4.101):

– ввести значение x для второй итерации, получаемой по формуле $x_2 = x_1 + h = 1 + 0{,}1 = 1{,}1$ (C4) (рис. 4.98);

– вычислить коэффициенты k_1, k_2, k_3, k_4 , для этого необходимо выделить формулы, находящиеся в ячейках диапазона B5 : B8 (рис. 4.99) и скопировать их в соседние ячейки C5 : C8 (рис. 4.100);

– вычислить значение y для следующей итерации, для этого необходимо скопировать формулу из ячейки B9 в ячейку C9 (рис. 4.101).

C4 =B4+$D$1

	A	B	C	D
1	**Шаг изменения x**			0,1
2				
3	**Итерация**	**1**	**2**	
4	**x**	0,00	0,10	
5	**k_1**	3,00		
6	**k_2**	3,35		
7	**k_3**	3,39		
8	**k_4**	3,78		
9	**y**	1,00	1,34	
10				

Рис. 4.98

	A	B	C	D
1	**Шаг изменения x**			0,1
2				
3	**Итерация**	**1**	**2**	
4	**x**	0,00	0,10	
5	**k_1**	3,00		
6	**k_2**	3,35		
7	**k_3**	3,39		
8	**k_4**	3,78		
9	**y**	1,00	1,34	
10				

Рис. 4.99

	A	B	C	D
1	**Шаг изменения x**			0,1
2				
3	**Итерация**	**1**	**2**	
4	**x**	0,00	0,10	
5	**k_1**	3,00	3,78	
6	**k_2**	3,35	4,22	
7	**k_3**	3,39	4,26	
8	**k_4**	3,78	4,75	
9	**y**	1,00	1,34	
10				

Рис. 4.100

	A	B	C	D
1	**Шаг изменения x**			0,1
2				
3	**Итерация**	**1**	**2**	**3**
4	**x**	0,00	0,10	
5	**k_1**	3,00	3,78	
6	**k_2**	3,35	4,22	
7	**k_3**	3,39	4,26	
8	**k_4**	3,78	4,75	
9	**y**	1,00	1,34	1,76
10				

Рис. 4.101

Шаг 4. Повторять действия, описанные в *шаге 3*, пока x не станет равен 2. Для этого необходимо:

– вычислить значения x и коэффициентов k_1, k_2, k_3, k_4, для этого необходимо выделить формулы, записанные в ячейках C3 : C8, и растянуть маркер заполнения вправо, пока значение x не достигнет значения 2;

– вычислить значения y для всех итераций, для чего скопировать формулу ячейки C9 в ячейки D9 : W9.

Результаты вычислений представлены на рис. 4.102.

	A	B	C	D	E	F	G	O	T	U	V	W
1	**Шаг изменения x**			0,1								
2												
3	**Итерация**	**1**	**2**	**3**	**4**	**5**	**6**	**…**	**18**	**19**	**20**	**21**
4	**x**	0,00	0,10	0,20	0,30	0,40	0,50	…	1,70	1,80	1,90	2,00
5	**k_1**	3,00	3,78	4,75	5,94	7,41	9,22	…	114,38	140,34	172,11	210,99
6	**k_2**	3,35	4,22	5,28	6,60	8,23	10,23	…	126,10	154,68	189,67	232,47
7	**k_3**	3,39	4,26	5,34	6,67	8,31	10,33	…	127,27	156,12	191,42	234,62
8	**k_4**	3,78	4,75	5,94	7,41	9,23	11,46	…	140,41	172,20	211,10	258,70
9	**y**	1,00	1,34	1,76	2,29	2,96	3,79	…	54,45	67,14	82,71	101,80
10												

Рис. 4.102

4.5.2. Решение дифференциального уравнения методом Эйлера

***Задача** 4.12.* Решить дифференциальное уравнение $y' = 2y + e^x$ на отрезке [0, 2] методом Эйлера. Начальное условие Коши $y(0) = 1$.

Алгоритм решения

Шаг 1. Распишем формулы для вычисления коэффициентов и значения уравнения.

Для $i + 1$ итерации значение $y_{i+1} = y_i + hf(x, y)$.

Шаг изменения переменной x ($x_{i+1} - x_i$) установить равным 0,1 (ячейка B1).

Шаг 2. Первая итерация:

– ввести значение шага для переменной x в ячейку B1, ссылку на которую в формулах следует сделать абсолютной (обозначить ее $B$1), таким образом при копировании формул ссылка на эту ячейку не сместится;

– ввести начальные значения x, y (B4, B5) (рис. 4.103).

Шаг 3. Вторая итерация:

– ввести значение x_2, получаемое по формуле $x_2 = x_1 + h = 1 + 0{,}1 = 1{,}1$ (C4) (рис. 4.104);

	A	B	C
1	**Шаг h**	0,1	
2			
3	**Итерация**	**1**	
4	**x**	0,00	
5	**y**	1,00	
6			

Рис. 4.103

C4 =B4+$B$1

	A	B	C	D
1	**Шаг h**	0,1		
2				
3	**Итерация**	**1**	**2**	
4	**x**	0,00	0,10	
5	**y**	1,00		
6				

Рис. 4.104

– вычислить значение y для следующей итерации (C5) по формуле $y_2 = y_1 + h(2 \cdot y_1 + e^{x_1})$ (рис. 4.105).

Шаг 4. Повторять действия, описанные в *шаге 3*, пока x не станет равен 2. Для этого необходимо выделить диапазон ячеек C3 : C5 (рис. 4.106) и растянуть маркер заполнения вправо. Результаты вычислений приведены на рис. 4.107.

C5 =B5+$B$1*(2*B5+EXP(B4))

	A	B	C	D	E	F
1	**Шаг h**	0,1				
2						
3	**Итерация**	**1**	**2**			
4	**x**	0,00	0,10			
5	**y**	1,00	1,30			
6						

Рис. 4.105

	A	B	C	D
1	**Шаг h**	0,1		
2				
3	**Итерация**	**1**	**2**	
4	**x**	0,00	0,10	
5	**y**	1,00	1,30	
6				

Рис. 4.106

	A	B	C	D	E	F	G	S	T	U	V	W
1	**Шаг h**	0,1										
2												
3	**Итерация**	**1**	**2**	**3**	**4**	**5**	**6**	**…**	**18**	**19**	**20**	**21**
4	**x**	0,00	0,10	0,20	0,30	0,40	0,50	…	1,70	1,80	1,90	2,00
5	**y**	1,00	1,30	1,67	2,13	2,69	3,37	…	39,81	48,32	58,59	70,97
6												

Рис. 4.107

4.5.3. Решение дифференциальных уравнений с частными производными методом конечных разностей

Задача 4.13. Провести моделирование температурного поля сечения детали на основе уравнения теплопроводности при заданных начальных и граничных условиях $\frac{\partial T}{\partial t} = a\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$. Конфигурация сечения отливки представлена на рис. 4.108. Темно-серым цветом представлено сечение отливки, белым – форма, светло-серым – окружающая среда.

20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	500	500	20	20	20	20	20
20	20	20	20	500	500	500	500	20	20	20	20
20	20	20	20	20	500	500	20	20	20	20	20
20	20	500	500	500	500	500	500	500	500	20	20
20	20	20	500	500	500	500	500	500	20	20	20
20	20	20	500	500	20	20	500	500	20	20	20
20	20	20	500	500	20	20	500	500	20	20	20
20	20	500	500	500	500	500	500	500	500	20	20
20	20	20	20	500	500	500	500	20	20	20	20
20	20	20	20	500	500	500	500	20	20	20	20
20	20	20	20	20	500	500	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20
20	20	20	20	20	20	20	20	20	20	20	20

Рис. 4.108

Заданы теплофизические параметры для разных типов клеток:

– для клеток формы: теплоемкость c = 519 Дж/К, теплопроводность l = 44,4 Вт/(м·К), плотность r = 1600 кг/м^3, начальная температура t = 20 °С;

– для клеток отливки: теплоемкость c = 343 Дж/К, теплопроводность l = 110 Вт/(м·К), плотность r = 8900 кг/м^3, начальная температура t = 500 °С.

Шаг по пространству равен 0,002 м и шаг по времени – 0,0005 с.

Алгоритм решения

Шаг 1. В виде исходного условия необходимо использовать уравнение теплообмена для двумерной системы:

$$\frac{\partial T}{\partial t} = a\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right).$$

Численное решение уравнения теплопроводности записывается в конечно-разностном представлении в виде

$$T_{i,j}^{n+1} = T_{i,j}^{n} + a\Delta t\left(\frac{T_{i+1,j}^{n} + T_{i-1,j}^{n} + T_{i,j+1}^{n} + T_{i,j-1}^{n} - 4T_{i,j}^{n}}{\Delta x^2}\right), \qquad (4.12)$$

где $T_{i,j}$ – температура клетки i-й строки и j-го столбца по осям x, y соответственно;

n, $n + 1$ – переменные временного слоя;

a – коэффициент температуропроводности:

$$a = \frac{\lambda}{c \cdot \rho},$$

где λ – теплопроводность,

c – теплоемкость,

ρ – плотность материала клетки.

Шаг 2. Создать объект моделирования. Для этого необходимо выделить диапазон ячеек A2 : L15, заполнить клетки цифрами, как показано на рис. 4.109, выделить границы клеток диапазона с помощью кнопки *Границы* на панели *Форматирование*.

	A	B	C	D	E	F	G	H	I	J	K	L
1	20	20	20	20	20	20	20	20	20	20	20	20
2	20	20	20	20	20	20	20	20	20	20	20	20
3	20	20	20	20	20	500	500	20	20	20	20	20
4	20	20	20	20	500	500	500	500	20	20	20	20
5	20	20	20	20	20	500	500	20	20	20	20	20
6	20	20	500	500	500	500	500	500	500	500	20	20
7	20	20	20	500	500	500	500	500	500	20	20	20
8	20	20	20	500	500	20	20	500	500	20	20	20
9	20	20	20	500	500	20	20	500	500	20	20	20
10	20	20	500	500	500	500	500	500	500	500	20	20
11	20	20	20	20	500	500	500	500	20	20	20	20
12	20	20	20	20	500	500	500	500	20	20	20	20
13	20	20	20	20	20	500	500	20	20	20	20	20
14	20	20	20	20	20	20	20	20	20	20	20	20
15	20	20	20	20	20	20	20	20	20	20	20	20

Рис. 4.109

Шаг 3. Ввести температурную шкалу в представленном на рис. 4.110 виде. В соответствии с этой шкалой клетки отливки будут окрашены в тот или иной цвет в зависимости от значения температуры клетки.

O	P
Шкала температур	
Температура	**Цвет клетки**
0-124	
125-250	
251-375	
376-500	

Рис. 4.110

Шаг 4. Сделать первоначальную заливку обеъкта моделирования. Для этого необходимо:

– выделить диапазоны ячеек A1 : A15, A1 : L1, L1 : L15, A15 : L15 – область окружающей среды и сделать заливку зеленым цветом с помощью инструмента *Цвет заливки* на панели *Форматирование*;

– выделить диапазон B2 : K14 и сделать заливку фиолетовым цветом.

В результате первоначальной заливки изображение детали будет иметь вид, представленный на рис. 4.111.

Шаг 5. Ввести исходные данные, а именно значения теплопроводности, теплоемкости, плотности, шагов по времени и пространству. Вычислить значение коэффициентов *a* для формы и отливки (см. рис. 4.111).

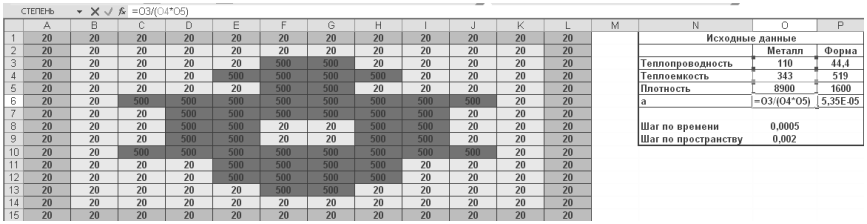


Рис. 4.111

Шаг 6. Для того чтобы выделить клетки отливки цветом, в зависимости от температуры клетки в соответствии с температурной шкалой, необходимо применить условное форматирование. Для этого необходимо выделить диапазон A2 : L15 и выбрать пункт меню *Формат* → *Условное форматирование*... Появится диалоговое окно «Условное форматирование» (рис. 4.112), в котором необходимо задать три следующих условия:

– первое условие: если значение клетки находится в интервале между 125 и 250, то клетку закрашивать желтым цветом, для задания цвета клетки надо нажать кнопку *Формат* и затем на вкладке *Вид* выделить цвет заливки ячейки, для ввода следующего условия нажать кнопку *А также>>*;

– второе условие: если значение клетки находится в интервале между 251 и 375, то клетку закрашивать оранжевым цветом;

– третье условие: если значение клетки находится в интервале между 375 и 500, то клетку закрашивать красным цветом.

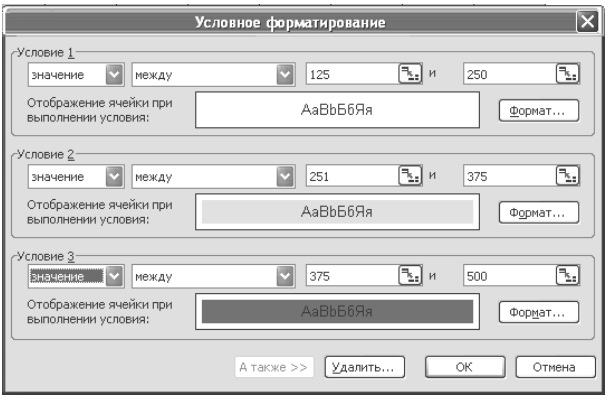


Рис. 4.112

Таким образом, если значение клетки не входит ни в один из описанных интервалов, т. е. значение температуры меньше 125, то она остается фиолетовой.

После выполнения условного форматирования изображение детали примет вид, представленный на рис. 4.113.

	A	B	C	D	E	F	G	H	I	J	K	L
1	20	20	20	20	20	20	20	20	20	20	20	20
2	20	20	20	20	20	20	20	20	20	20	20	20
3	20	20	20	20	20	500	500	20	20	20	20	20
4	20	20	20	20	500	500	500	500	20	20	20	20
5	20	20	20	20	20	500	500	20	20	20	20	20
6	20	20	500	500	500	500	500	500	500	500	20	20
7	20	20	20	500	500	500	500	500	500	20	20	20
8	20	20	20	500	500	20	20	500	500	20	20	20
9	20	20	20	500	500	20	20	500	500	20	20	20
10	20	20	500	500	500	500	500	500	500	500	20	20
11	20	20	20	20	500	500	500	500	20	20	20	20
12	20	20	20	20	500	500	500	500	20	20	20	20
13	20	20	20	20	20	500	500	20	20	20	20	20
14	20	20	20	20	20	20	20	20	20	20	20	20
15	20	20	20	20	20	20	20	20	20	20	20	20

Рис. 4.113

Шаг 7. Скопировать изображение получившейся детали ниже, новое изображение будет начинаться с ячейки A18.

Шаг 8. Вычислить температуру клеток формы в следующий момент времени, т. е. через 0,0005 с, для этого необходимо:

– ввести в ячейку B19 формулу (4.12) для вычисления температуры в следующий момент времени для формы, используя коэффициент *a* для формы (ячейка O6) (рис. 4.114); в качестве значения $T_{i,j}^{n}$ использовать значение ячейки B2, $T_{i+1,j}^{n}$ – ячейку C2, $T_{i-1,j}^{n}$ – ячейку A2, $T_{i,j+1}^{n}$ – ячейку B3, $T_{i,j-1}^{n}$ – ячейку B1; при вводе формулы необходимо обратить внимание на то, что адреса ячеек O6 (коэффициент *a*), O8 (шаг по времени), O9 (шаг по пространству) необходимо вводить как абсолютные ссылки;

– выделить ячейку и скопировать ее на все ячейки формы, в результате температурное поле детали будет иметь вид, представленный на рис. 4.114.

Шаг 9. Вычислить температуру клеток отливки в следующий момент времени, для этого необходимо:

– для вычисления температуры в следующий момент времени для отливки ввести в ячейку F20 формулу (4.12), используя коэффициент *a* для отливки (ячейка P6) (рис. 4.115); при вводе формулы

адреса ячеек P6 (коэффициент *a*), O8 (шаг по времени), O9 (шаг по пространству) необходимо вводить как абсолютные ссылки;

– выделить ячейку и скопировать ее на все ячейки отливки, в результате температурное поле детали через 0,0005 с будет иметь вид, представленный на рис. 4.115.

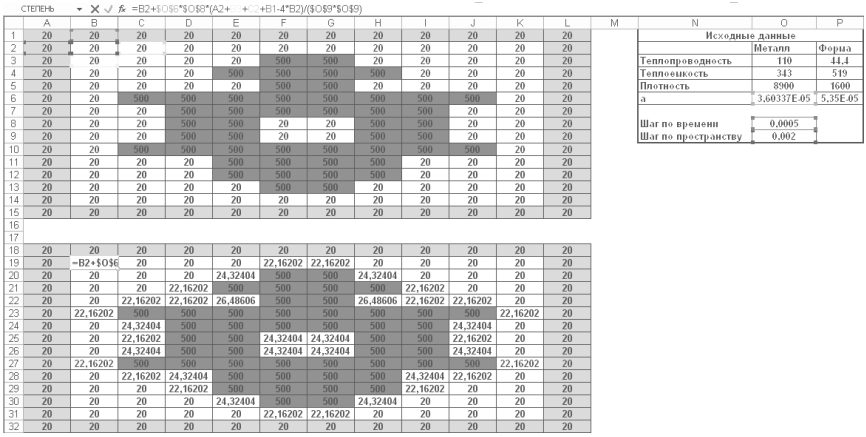


Рис. 4.114

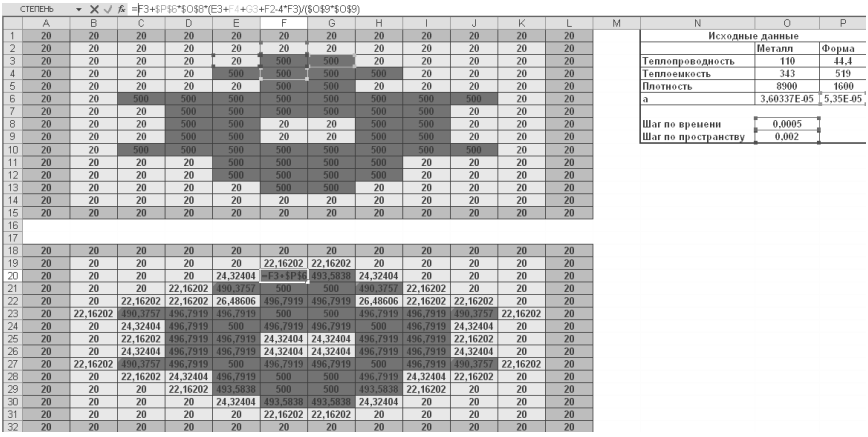


Рис. 4.115

Шаг 10. Для вычисления температуры клеток отливки и детали через $2 \cdot 0{,}0005 = 0{,}001$ с необходимо вставить копию изображения последнего температурного поля детали ниже – новое изображение

будет начинаться с ячейки A35. Температурное поле формы и отливки через 0,001 с будет иметь вид, представленный на рис. 4.116.

	A	B	C	D	E	F	G	H	I	J	K	L
34												
35	20	20	20	20	20	20	20	20	20	20	20	20
36	20	20	20	20	20,02921	24,26593	24,26593	20,02921	20	20	20	20
37	20	20	20	20,02921	28,49793	487,3396	487,3396	28,49793	20,02921	20	20	20
38	20	20	20,01948	24,25148	481,0954	499,8714	499,8714	481,0954	24,25148	20,01948	20	20
39	20	20,01948	24,25148	24,31933	32,79275	493,6915	493,6915	32,79275	24,31933	24,25148	20,01948	20
40	20	24,24174	481,0451	493,5768	493,6915	499,9357	499,9357	493,6915	493,5768	481,0451	24,24174	20
41	20	20,02921	28,52211	493,6556	499,9142	493,677	493,677	499,9142	493,6556	28,52211	20,02921	20
42	20	20,00974	24,30959	493,6197	493,6556	28,58023	28,58023	493,6556	493,6197	24,30959	20,00974	20
43	20	20,02921	28,52211	493,6342	493,6556	28,58023	28,58023	493,6556	493,6342	28,52211	20,02921	20
44	20	24,24174	481,0451	493,6127	499,9142	493,677	493,677	499,9142	493,6127	481,0451	24,24174	20
45	20	20,01948	24,26121	28,56075	493,6556	499,9571	499,9571	493,6556	28,56075	24,26121	20,01948	20
46	20	20	20,01948	24,27566	487,3611	499,9142	499,9142	487,3611	24,27566	20,01948	20	20
47	20	20	20	20,02921	28,51238	487,3396	487,3396	28,51238	20,02921	20	20	20
48	20	20	20	20	20,02921	24,26593	24,26593	20,02921	20	20	20	20
49	20	20	20	20	20	20	20	20	20	20	20	20

Рис. 4.116

Таким образом, для получения температурного поля детали через $0{,}0005 \cdot n$ с необходимо скопировать изображение n раз. Например, на рис. 4.117 показано температурное поле формы и отливки через $0{,}0005 \cdot 40 = 0{,}02$ с (изображение скопировано 40 раз).

	A	B	C	D	E	F	G	H	I	J	K	L
682	20	20.00638	20.10762	21.40434	32.50346	73.87182	73.87182	32.50346	21.40434	20.10762	20.00638	20
683	20	20.08786	21.17384	32.34375	114.4096	335.8911	335.8911	114.4096	32.34375	21.17384	20.08786	20
684	20	20.93033	28.42275	71.02797	291.5114	446.0744	446.0744	291.5114	71.02797	28.42275	20.93033	20
685	20	28.03056	69.67458	94.94278	181.2901	423.5314	423.5314	181.2901	94.94278	69.67458	28.03056	20
686	20	64.30547	272.3832	384.8715	421.7557	472.4094	472.4094	421.7557	384.8715	272.3832	64.30547	20
687	20	32.79804	124.1249	407.3907	463.9478	420.5348	420.5348	463.9478	407.3907	124.1249	32.79804	20
688	20	26.85741	91.81055	398.8435	411.6045	150.7111	150.7111	411.6045	398.8435	91.81055	26.85741	20
689	20	32.78544	123.8112	401.7035	411.7498	150.7383	150.7383	411.7498	401.7035	123.8112	32.78544	20
690	20	64.36684	273.5481	395.0755	462.0695	420.9028	420.9028	462.0695	395.0755	273.5481	64.36684	20
691	20	28.25754	73.5786	138.2039	411.7082	477.4103	477.4103	411.7082	138.2039	73.5786	28.25754	20
692	20	20.97188	29.17483	79.99401	346.3349	460.8269	460.8269	346.3349	79.99401	29.17483	20.97188	20
693	20	20.09148	21.24164	33.17909	119.6217	337.625	337.625	119.6217	33.17909	21.24164	20.09148	20
694	20	20.00659	20.1115	21.45337	32.81463	73.96317	73.96317	32.81463	21.45337	20.1115	20.00659	20
695	20	20	20	20	20	20	20	20	20	20	20	20

Рис. 4.117

4.6. **Решение задач оптимизации с использованием надстройки MS EXCEL «Поиск решения»**

Задача 4.14. Разработать алгоритм и определить с помощью приложения MS EXCEL комбинацию значений переменных x_1, x_2, x_3, изменяющихся в заданных интервалах, для системы нелинейных уравнений

$$\begin{cases} 4\sin x_1 + 50x_2 - 3\ln x_3 = 40 \to F_1, \\ 5\sin x_1 - 30\cos x_2 - \ln x_3 = 50 \to F_2, \\ 6\sin x_1 - 4\cos x_2 - 5\ln x_3 = 80 \to F_3, \end{cases}$$

$F_1(x_1, x_2, x_3) = 0$, $F_2(x_1, x_2, x_3) = 0$, $F_3(x_1, x_2, x_3) = 0$.

Функционал цели выбрать на основе условия

$$F = F_1^2 + F_2^2 + F_3^2 \to \min.$$

Предположить следующие интервалы изменения переменных: $x_1 = [-10; 10]$, $x_2 = [10; 40]$, $x_3 = [0; 50]$.

Алгоритм решения

Сущность решения задачи состоит в том, что строится функционал вида $F = F_1^2 + F_2^2 + F_3^2 \to \min$. При этом функции, входящие в него, являются уравнениями системы $F_1(x_1, x_2, x_3) = 0$, $F_2(x_1, x_2, x_3) = 0$, $F_3(x_1, x_2, x_3) = 0$. Решение ищется таким образом, чтобы минимизировать функционал.

Шаг 1. Ввести интервалы изменения значений переменных x_1, x_2, x_3 (рис. 4.118).

Шаг 2. Задать начальные значения переменных x_1, x_2, x_3 для поиска (ячейки F2, F3, F4) – это минимальные значения интервалов переменных (рис. 4.119).

	A	B	C	D
1			**Интервал значений**	
2		**x1**	-10	10
3		**x2**	10	40
4		**x3**	0	56

Рис. 4.118

F
Поиск
-10
10
0,1

Рис. 4.119

Шаг 3. Ввести обозначения для F_1, F_2, F_3:

$$\begin{cases} F_1 = 4\sin x_1 + 50x_2 - 3\ln x_3 - 40, \\ F_2 = 5\sin x_1 - 30\cos x_2 - \ln x_3 - 50, \\ F_3 = 6\sin x_1 - 4\cos x_2 - 5\ln x_3 - 80. \end{cases}$$

Шаг 4. Ввести формулы для вычисления критериев F_1, F_2, F_3 (ячейки F5, F6, F7) (рис. 4.120) и критерия F (ячейка F8) (рис. 4.121).

Рис. 4.120

Рис. 4.121

Шаг 5. Запуск поиска решения. Для этого надо выбрать пункт меню *Сервис* → *Поиск решения*… Появится диалоговое окно, в котором необходимо (рис. 4.122):

– в качестве целевой ячейки установить ячейку $F$8 – значение F;

– выбрать пункт *равной минимальному значению*, т. е. значения x_1, x_2, x_3 будут изменяться таким образом, чтобы значение F приняло минимальное значение;

– в поле *Изменяя ячейки* выбрать ячейку $F$2 : $F$4 – значения x_1, x_2, x_3;

– в поле списка *Ограничения* добавить ограничения на изменение ячейки $F$2 – значение x_1 должно входить в интервал изменения (должно быть больше минимального значения интервала, но меньше максимального), аналогично добавить ограничения на ячейки $F$3 (значение x_2) и $F$4 (значение x_3). Границы интервалов переменных находятся в ячейках C2 : D4.

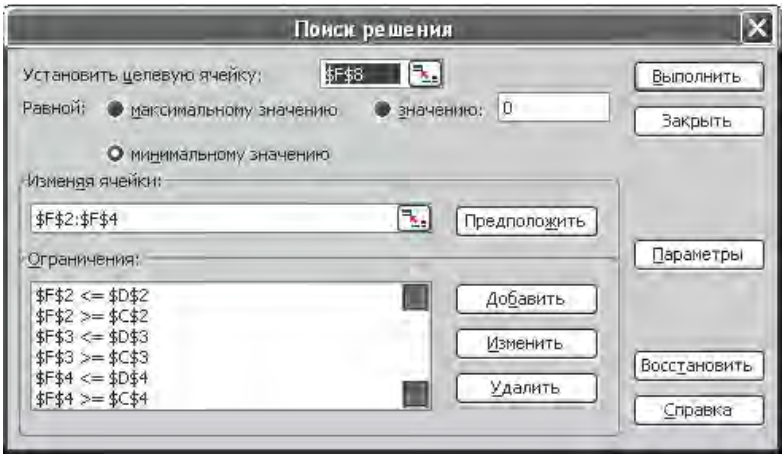


Рис. 4.122

После нажатия кнопки *Выполнить* будет выполнен поиск минимального значения *F* с заданными условиями. Затем после подтверждения в диалоговом окне *Результаты поиска решения* в ячейках F2, F3, F4 появятся значения x_1, x_2, x_3, при которых *F* (ячейка F8) принимает минимальное значение (рис. 4.123). Таким образом, минимальное значение *F* составляет 3178,004.

	A	B	C	D	E	F
1			**Интервал значений**			**Поиск**
2		**x1**	-10	10		-10
3		**x2**	10	40		11.07046774
4		**x3**	0.1	56		0.1
5					**F1**	-27.17498724
6					**F2**	-47.22201317
7					**F3**	14.47775829
8					**F**	3178.003945

Рис. 4.123

5. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Оформление заданий

Курсовой проект состоит из следующих частей:
1) постановка задачи;
2) алгоритм решения задачи с пояснениями;
3) программа решения задачи с комментариями;
4) результаты расчетов (можно частично);
5) графики и рисунки, если необходимо;
6) выводы.

5.1. **Задачи по разработке алгоритмов и программ с использованием разветвляющихся процессов**

Постановка задачи. Разработать алгоритм и программу для вычисления функции $Z(x)$, заданной интервально на различных промежутках. Если функция $Z(x)$ не определена при каких-либо значениях x, то в программе необходимо предусмотреть вывод сообщения «При x = … функция не определена» и вывести значение x. Общий вид задания функции

$$Z(x)=\begin{cases} f_0(x), \text{если } x \le x_0; \\ f_1(x), \text{если } x_0 < x \le x_1; \\ f_2(x), \text{если } x_1 < x \le x_2; \\ \dots \\ f_N(x), \text{если } x_{N-1} < x \le x_N; \\ 0, \quad \text{если } x > x_N. \end{cases}$$

Функция $f_i(x)$ представлена в табл. 5.1, а варианты заданий – в табл. 5.2.

Значения a, b, c, d вычисляются по формулам

$$a=\sum_{j=1}^{n+3}\sin j,\ b=\sum_{j=1}^{n+3}\frac{1}{\ln j},\ c=\sum_{j=1}^{n+3}(j^3+j),\ d=a+b+c,$$

где n – номер варианта.

Таблица 5.1

Индекс i-й функции	$f_i(x)$
0	$e^{ax} + 2\sin^2(bx) + \cos^{3,4}(cx) + d \cdot \mathrm{tg}(x)$
1	$e^{-bax} + \cos(b\sqrt[7]{x^8}) + \frac{1}{101}\ln\lvert cx\rvert + dx^4$
2	$\ln\lvert\ln\lvert ax\rvert\rvert + cbx^{3/4} + d\ln^3(x) - \sqrt{xa}$
3	$\ln(xd) + \cos(ax)/\mathrm{tg}(bx) + \sqrt{(x+7)/c} + (x+7)b$
4	$b\sqrt{x} + \sqrt{\ln(ax)} + cx^3 + dx^{1/10}$
5	$\ln^4(ax) + x^{3,4}b^2 + d^3x^4 - \sqrt{c^3}\cos(x)$
6	$ce^{-4x} + a\ln(x+50) + b\sqrt{(x+70)^4} + d^3\ln x/(x^3+1)$
7	$a\cos^3\lvert x\rvert + b\sin(dx) + \mathrm{ctg}^4(x) + d\ln^2(x)$
8	$a\mathrm{tg}(x) + b\ln(\sin(x)) + c\ln(\mathrm{tg}(x)) + d^3\cos(x)$
9	$\sqrt[3]{a/\cos(x)} + e^{bx} + \mathrm{tg}(cx^2)\sin^2(cx) - d \cdot x^{3/4}$
10	$ae^{-x} + \cos(bx) + \frac{1}{5x}\ln(dx) + ax^4$
11	$\ln\lvert ax\rvert + b\cos(x^4)\sin(x^{-3}) - c\ln^3(x) + \sqrt{x/d}$
12	$a\ln(xa) + \cos^2(bx)/\mathrm{tg}(bx) + \sqrt{(x-7)/c} + (x+7)^3 d$
13	$a\sqrt{\ln(x)} + \sqrt{\mathrm{tg}(bx)} - cx^{4/5} + dx^3/\cos(x)$
14	$\ln\ (\sqrt{a}x) + x^3b^2 + \sin(cx^4) - \sqrt{d^3}\mathrm{tg}(\sqrt{x})$
15	$ae^{-4x} + b\ln(x-5x^3) + c\sqrt{(x+c)^3} + \ln(d)/(x^3-2d)$
16	$\sqrt{a}\cos^3\lvert x\rvert + (1/b)\sin(4x) + \mathrm{ctg}^3(cx) + d\ln^2(2x)$
17	$a\cos^{3/4}(x)/d + b\sqrt{\sin(\ln b^2 x)} + \ln(c \cdot \mathrm{tg}(x)) + \cos(x^{d/2})$
18	$ax^5 - \sqrt{\cos(b)}\sin\ (bx) + ce^{-cx}x + x^{-2d}$

Окончание табл. 5.1

Индекс i-й функции	$f_i(x)$
19	$e^{-bx} + \cos(ax - ax^2) + \ln(\sqrt{\frac{cx}{x^3}}) + \sin(dx^2)$
20	$ax^{3a} + b\ln^3(x) - \sqrt{dx} + c\ln(x^{-c})$
21	$\ln(ax) + \cos(x^b)/\ln(b^x) - (x + 7d) + c\frac{e^{2cx}}{\ln(cx^{2/7})}$
22	$a\sqrt{x^a} + \cos^{b/2}(x^3) + \ln(c \cdot \mathrm{tg}(x)) + d\sqrt{\frac{e^x}{x^d}}$
23	$\ln\ (\cos(ax)) + b^2 e^{bx} - cx^5 + \sqrt{d^3}\ln\lvert x\rvert$
24	$e^{-cx/3} + a\ln(ax) + b\sqrt{b^4\cos^2(x)} - d\ \ /(x^3 + 7)$
25	$a\ln^{3/8}\lvert x\rvert + b\sin(be^{-b/x}) + (x - c) + d\frac{\mathrm{tg}^2(x^3)}{\ln(x - d)}$

Таблица 5.2

Номер варианта	Комбинация интервальной функции $Z(x)$	Формула для вычисления ограничений $Z(x)$
0	f_0, f_2, f_4, f_6, f_8	$x_i = 10i - 20$
1	f_1, f_3, f_5, f_7, f_9	$x_i = 3i - 10$
2	$f_2,\ f_5,\ f_8,\ f_{11},\ f_{14}$	$x_i = 4i - 1{,}2$
3	$f_4,\ f_7,\ f_{11},\ f_{15},\ f_{25}$	$x_i = 5i - 80$
4	$f_0,\ f_4,\ f_8,\ f_{12},\ f_{16}$	$x_i = 7i - 99$
5	$f_1,\ f_5,\ f_9,\ f_{13},\ f_{17}$	$x_i = 6i - 30$
6	$f_{10},\ f_{12},\ f_{14},\ f_{16},\ f_{18}$	$x_i = 7i - 11$
7	$f_{15}, f_{16}, f_{17}, f_{18}, f_{19}$	$x_i = 3i - 12$
8	$f_{20}, f_{21}, f_{23}, f_{24}, f_{25}$	$x_i = i - 12$
9	$f_0, f_6, f_{12}, f_{18}, f_{22}$	$x_i = 6i - 34$

Окончание табл. 5.2

Номер варианта	Комбинация интервальной функции $Z(x)$	Формула для вычисления ограничений $Z(x)$
10	$f_2, f_5, f_8, f_{19}, f_{24}$	$x_i = 9i - 22$
11	$f_2,\ f_7,\ f_8,\ f_{20},\ f_{23}$	$x_i = 5i - 6$
12	$f_1,\ f_2,\ f_{14},\ f_{16},\ f_{18}$	$x_i = 2i - 9$
13	$f_{11},\ f_{14},\ f_{17},\ f_{21},\ f_{25}$	$x_i = 7i - 45$
14	$f_{10}, f_{16}, f_{20}, f_{22}, f_{23}$	$x_i = 11i - 14$
15	$f_0, f_6, f_7, f_{12}, f_{13}$	$x_i = 21i - 100$
16	$f_3, f_8, f_{13}, f_{14}, f_{18}$	$x_i = 15i - 27$
17	$f_9, f_{10}, f_{15}, f_{20}, f_{21}$	$x_i = 3i - 11$
18	$f_5, f_7, f_9, f_{19}, f_{24}$	$x_i = i - 2$
19	$f_4, f_6, f_{18}, f_{21}, f_{23}$	$x_i = 12i - 63$
20	$f_2, f_{14}, f_{16}, f_{18}, f_{25}$	$x_i = 7i - 9$
21	$f_3, f_4, f_5, f_{10}, f_{20}$	$x_i = 25i - 41$
22	$f_9, f_{10}, f_{19}, f_{20}, f_{24}$	$x_i = 2i - 29$
23	$f_0, f_5, f_{10}, f_{12}, f_{13}$	$x_i = 13i - 78$
24	$f_2, f_3, f_4, f_{17}, f_{18}$	$x_i = 8i - 42$
25	$f_6, f_{13}, f_{16}, f_{22}, f_{24}$	$x_i = 3i - 55$

Таким образом, студент, выполняющий 4-й вариант, должен разработать алгоритм и программу для вычисления функции $Z(x)$:

$$Z(x) = \begin{cases} f_0, & x \le x_0, & x_0 = 7 \cdot 0 - 99 = -99, \\ f_4, & x_0 < x \le x_4, & x_4 = 7 \cdot 4 - 99 = -71, \\ f_8, & x_4 < x \le x_8, & x_8 = 7 \cdot 8 - 99 = -43, \\ f_{12}, & x_8 < x \le x_{12}, & x_{12} = 7 \cdot 12 - 99 = -15, \\ f_{16}, & x_{12} < x \le x_{16}, & x_{16} = 7 \cdot 16 - 99 = 13, \\ 0, & x > x_{16}, & \end{cases}$$

где $f_0 = e^{ax} + 2\sin^2 bx + \cos^{3,4} cx + d \cdot \operatorname{tg} x$,

$f_4 = b\sqrt{x} + \sqrt{\ln(ax)} + cx^3 + dx^{\frac{1}{10}}$,

$f_8 = a\operatorname{tg} x + b\ln(\sin x) + c\ln(\operatorname{tg}(x)) + d^3 \cos x$,

$f_{12} = a\ln(xa) + \cos^2(bx) / \operatorname{tg}(bx) + \sqrt{(x-7)/c} + (x+7)^3 d$,

$f_{16} = \sqrt{a}\cos^3|x| + (1/b)\sin(4x) + \operatorname{ctg}^3 cx + d\ln^2 2x$.

Значения параметров вычисляются как $a = \sum_{j=1}^{7} \sin j$, $b = \sum_{j=1}^{7} \frac{1}{\ln j}$,

$c = \sum_{j=1}^{7} (j^3 + j)$, $d = a + b + c$.

5.2. **Задачи по разработке алгоритмов и программ с использованием циклических процессов**

Постановка задачи. Разработать алгоритм и программу вычисления функции $C(x_1, x_2, x_3, x_4)$, зависящей от четырех переменных x_1, x_2, x_3, x_4, которые изменяются по различным законам. Значения переменных x_1, x_2, x_3, x_4 задаются на основе формул

$$x_1 = 5n;\ 25n\ (0{,}5n);$$

$$x_2 = n + 1;\ n + 10\ (0{,}1n);$$

$$x_3 = 8 + n;\ n^2;\ n^2 + 3;\ n^2 + 0{,}4;$$

$$x_4 = n - 1;\ (n-2)^2;\ n + 0{,}8;\ n + 20;$$

где n – вариант задания.

Вид функции $C(x_1, x_2, x_3, x_4)$ выбирается из табл. 5.1 (i – вариант) при условии $a = 1$, $b = 1$, $c = 1$, $d = 1$, где каждое из слагаемых зависит от своей переменной. В программе необходимо предусмотреть вычисление значений $C(x_1, x_2, x_3, x_4)$ для всех комбинаций x_1, x_2, x_3, x_4. *Обязательно предусмотреть полный перебор* x_1, x_2, x_3, x_4. Вычислить сумму всех $C(x_1, x_2, x_3, x_4)$ и вывести ее на монитор. В про-

грамме организовать вывод x_1, x_2, x_3, x_4, для которых выполняется условие

$$-(20n + 1) \le C(x_1, x_2, x_3, x_4) \le (20n^2 + 1).$$

Таким образом, для 4-го варианта необходимо написать программу по вычислению функции

$$C(x_1, x_2, x_3, x_4) = \sqrt{x_1} + \sqrt{\ln(x_2)} + x_3^{\,3} + x_4^{\,1/10},$$

где $x_1 = 5 \cdot 4$; $25 \cdot 4$ $(0{,}5 \cdot 4) = 20$; 100 (2);
$x_2 = 4 + 1$; $4 + 10$ $(0{,}1 \cdot 4) = 5$; 14 (0,4);
$x_3 = 8 + 4$; 16; $16 + 3$; $16 + 0{,}4 = 12$; 16; 19; 16,4;
$x_4 = 4 - 1$; $(4 - 1)^2$; $4 + 0{,}8$; $4 + 20 = 3$; 9; 4,8; 24.

Таким образом, переменные x_1 и x_2 заданы с равномерными шагами 2 и 0,4, а x_3 и x_4 – массивами из четырех значений. Вывод $C(x_1, x_2, x_3, x_4)$ при условии

$$-(20 \cdot 4 + 1) \le C(x_1, x_2, x_3, x_4) \le (20 \cdot 4^2 + 1).$$

Внимание. Если значение функции $C(x_1, x_2, x_3, x_4)$ не определено при какой либо комбинации x_1, x_2, x_3, x_4, то на экран монитора вывести x_1, x_2, x_3, x_4, при которых $C(x_1, x_2, x_3, x_4)$ не определена, и вывести фразу «C – не определена».

5.3. **Задачи по разработке алгоритмов и программ для обработки матриц**

Постановка задачи. Разработать алгоритм и программу для нахождения максимального и минимального элементов матрицы Z размером 30×30 и их индексов (i, j). Экстремальные значения Z следует искать среди элементов, для которых выполняются условия

$$\begin{cases} -0{,}3 < z_{ij} < 0{,}3; \\ 10 < i + j < 50. \end{cases}$$

Элементы матрицы z_{ij} формируются исходя из условий

$$z_{ij} = \begin{cases} \sin(f_n), & \text{если} \quad i > j, \\ \cos(f_{n+1}), & \text{если} \quad i \le j, \end{cases}$$

где n – номер варианта;

f_n, f_{n+1} – функции, взятые из табл. 5.1 в соответствии с вариантом n.

Функция f_n зависит от x, a, b, c, d (см. табл. 5.1), которые определяются как $x = n(i + j)$, $a = n \cdot i$, $b = n(i + 1)$, $c = n \cdot j$, $d = n(j + 1)$, где n – номер варианта. Если значение $f_n(x)$ не определено при заданных x, a, b, c, d, то $z_{ij} = 0$.

Таким образом, для 4-го варианта задача конкретизируется следующим образом. Разработать алгоритм и программу нахождения максимального и минимального элементов матрицы Z:

$$z_{ij} = \begin{cases} \sin(b\sqrt{x} + \sqrt{\ln(ax)} + cx^3 + dx^{\frac{1}{10}}), \text{ если } i > j, \\ \cos(\ln^4(ax) + x^{3.4}b^2 + d^3x^4 - \sqrt{c^3}\cos(x)), \text{если } i \le j, \end{cases}$$

$$x = 4(i + j),\ a = 4i,\ b = 4(i + 1),\ c = 4j,\ d = 4(j + 1).$$

В программе организовать вывод максимального и минимального элементов, а также их индексов i и j в файл *a.lst*.

5.4. **Задачи по разработке алгоритмов и программ для вычислений сложных сумм**

Постановка задачи. Разработать алгоритм и программу вычисления функции $Z(x) = \sin[S \cdot F(x)]$ в интервале $x \in [-100, 100]$ с шагом $\Delta x = 0{,}5$. Величина S определяется как сумма, заданная в табл. 5.3, а $y_i = i^2 + i + 1$. Функция $F(x)$ берется из табл. 5.1 в соответствии с вариантом задания. Значения a, b, c, d определяют как $a = nx$,

$b = (n + 1)x$, $c = \cos(nx)$, $d = \sin(nx)$. В программе организовать вывод x, $Z(x)$ в соответствии с условиями, накладываемыми на $Z(x)$. Если выполняется условие 1: $\cos(n) < Z(x) < \sin(n + 1)$, то x и $Z(x)$ выводятся в файл *An.l*st (n – вариант задания). Если выполняется условие 2: $\cos(n^2) < Z(x) < \sin(n^2 + 1)$, то x и $Z(x)$ выводятся в файл *Bn.l*st. Если выполняются одновременно оба условия, то вывод x и $Z(x)$ производится в файл *Cn.l*st. Если не выполняется ни одно из условий, то вывод x и $Z(x)$ происходит в файл *Dn.l*st. При выводе можно ограничиться двадцатью значениями x и $Z(x)$ в каждом из файлов.

Таким образом, для 4-го варианта необходимо вычислить функцию

$$Z(x) = \sin\left(S\left[b\sqrt{x} + \sqrt{\ln(ax)} + cx^3 + dx^{\frac{1}{10}} \right] \right),$$

где $a = 4x$, $b = 5x$, $c = \cos(4x)$, $d = \sin(4x)$.

Величина S определена как

$$S = \sum_{i=1}^{10} y_i \sum_{i=1}^{20} y_i - \sum_{i=1}^{10} (y_i^2 - 2y_i), \text{ где } y_i = i^2 + i + 1.$$

Определить $Z(x)$ в интервале $x = -100; 100$ (0,5). Запись $Z(x)$, x в файлы по условиям

$$\cos(4) < Z(x) < \sin(5) \Rightarrow An.lst;$$

$$\cos(16) < Z(x) < \sin(18) \Rightarrow Bn.lst;$$

$$\cos(4) < Z(x) < sin(5) \text{ и } \cos(4) < Z(x) < \sin(5) \Rightarrow Cn.lst;$$

$$\text{невыполнение условий} \Rightarrow Dn.lst.$$

Если функция $Z(x)$ не определена для данного x, то принять $Z(x) = 1$.

Таблица 5.3

Вариант	Вид функции	Вариант	Вид функции
0	$\sum_{i=1}^{100}\sin(y_i)/\sum_{i=1}^{10}(y_i^2-y_i^4)$	14	$\sum_{i=1}^{25}\ln(y_i)+\cos(\sum_{i=1}^{10}(y_i^4))$
1	$\sum_{i=1}^{50}(y_i\ln(\frac{1}{2+y_i}))/(\sum_{i=1}^{30}(y_i)/30)$	15	$\sum_{i=1}^{30}\sum_{i=1}^{40}\cos(y_i)\sin(y_j^3)$
2	$\sum_{i=1}^{100}y_i\sin(y^2{}_i)/\sum_{i=1}^{30}(y_i^7-y_i^2)$	16	$\sum_{i=1}^{10}(y_i-\ln(y_i)/\sum_{i=1}^{10}(\cos(y_i+y_i^2))$
3	$\sum_{i=1}^{100}\cos(1/y_i)+\sum_{i=1}^{20}\sum_{j=1}^{10}(y_i^2+y_j)$	17	$\sum_{i=1}^{70}\sum_{j=1}^{30}(y_i\,y_j)^{2/3}+\cos(\sum_{i=15}^{25}5\ln(y_i))$
4	$\sum_{i=1}^{10}y_i\sum_{i=1}^{20}y_i-\sum_{i=1}^{40}(y_i^2-2y_i)$	18	$\sum_{i=1}^{40}\sum_{j=1}^{20}(y_j^{\frac{2}{3}}y_i)+\frac{\sum_{i=1}^{10}(y_i^3-y_i^2-y_i)}{5}$
5	$\sum_{i=1}^{40}\sum_{j=1}^{20}(y_j^{\frac{2}{3}}y_i)+\frac{\sum_{i=1}^{10}(y_i^3-y_i^2-y_i)}{5}$	19	$\sum_{i=1}^{100}\cos^{4/5}(y_i)/\sum_{i=1}^{10}(18y_i+\mathrm{tg}(y_i^4))$
6	$\sum_{i=1}^{30}[\sin^2(y_i)-\cos^2(5y_i)]+\sum_{i=1}^{10}\ln(y_i^4-y_i^2)$	20	$\sum_{i=1}^{22}\sum_{i=1}^{30}(\lvert y_i^3-y_i^5\rvert)+\sum_{i=1}^{43}(\cos(y_i)-\sin(y_i))$

Окончание таблицы 5.3

Вариант	Вид функции	Вариант	Вид функции
7	$\sum_{i=1}^{30}\sum_{j=1}^{50}(y_j - y_j^2 - y_i)$	21	$\sum_{i=1}^{100}\frac{\sum_{j=1}^{10}(y_j^2 - y_j^4)}{y_i}$
8	$\sum_{i=1}^{35}({y_i}^5 + \sin(y_i)) / \sum_{i=1}^{10}\sum_{j=2}^{15}(y_i^2 y_j)$	22	$\sum_{i=1}^{10}(y_i - {y_i}^3 + y_i^4) / (\sum_{i=1}^{10}(y_i^3) - \sum_{i=1}^{25}\frac{y_i}{y_i^2})$
9	$\sum_{i=1}^{10}\sum_{j=1}^{50}({y_j}^2 - y_i) - \sum_{i=10}^{45}\cos(y_i^{2/3})$	23	$\sum_{i=1}^{40}(y_i) / \sum_{i=1}^{10}y_i^2 \sum_{i=10}^{20}y_i \sum_{i=30}^{45}y_i^3$
10	$\sum_{i=1}^{40}\sum_{j=1}^{20}(y_j y_i) - \sum_{i=1}^{10}(y_i^3 - y_i^2 - y_i)$	24	$\sum_{i=1}^{15}\ln(y_i) + \cos(\sum_{i=1}^{10}y_i^4)$
11	$\sum_{i=1}^{10}\sum_{j=1}^{15}(\lvert y_j - y_i \rvert + (y_i - \frac{1}{30}\sum_{i=1}^{10}y_i))$	25	$\sum_{i=1}^{20}\ln(\frac{y_i}{{y_i}^2}) - \sum_{i=1}^{30}(\frac{1}{y_i})$
12	$\sum_{i=1}^{10}\sum_{j=1}^{20}(y_j y_i) - \sum_{i=1}^{30}(y_i + y_i^2)$	26	$\sum_{i=1}^{70}(y_i - {y_i}^3) + \sum_{i=1}^{15}e^{-y_i}$
13	$\sum_{i=1}^{20}\sin(y_i) / \sum_{i=10}^{40}\cos(y_i^2 - y_i^4)$		

5.5. **Задачи по разработке алгоритмов и программ для решения нелинейных уравнений**

Постановка задачи. Разработать алгоритм и программу для решения нелинейного уравнения (табл. 5.4) с параметрами методом дихотомии. С помощью программы вычислить все корни уравнений в соответствии с указанным вариантом и заданной точностью $\varepsilon = 10^{-4}$. Параметр a задать самостоятельно методом перебора. Определить для каких значений параметра a уравнение имеет 1, 2, 3, 4, 5 действительных корней.

Таблица 5.4

Номер варианта	Уравнение
0	$e^{-ax} + 2\sin^2(ax) + \cos^{3,4}(ax) + a \cdot \mathrm{tg}(x) + a = 0$
1	$e^{-ax} + \cos(a\sqrt[7]{x^8}) + \frac{1}{101}\ln\lvert ax\rvert + ax^4 + a = 0$
2	$\ln\lvert\ln\lvert ax\rvert\rvert + ax^{3/4} + a\ln^3(x) - \sqrt{xa} + a = 0$
3	$\ln(xa) + \cos(ax)/\mathrm{tg}(ax) + \sqrt{(x+7)/a} + (x+7)a + a = 0$
4	$a\sqrt{x} + \sqrt{\ln(ax)} + ax^3 + ax^{1/10} + a = 0$
5	$\ln^4(ax) + x^{3.4}a^2 + a^3x^4 - \sqrt{a^3}\cos(x) + a = 0$
6	$ae^{-4x} + a\ln(x+50) + a\sqrt{(x+70)^4} + + a^3\ln(x)/(x^3+1) + a = 0$
7	$a\cos^3\lvert x\rvert + a\sin(ax) + a\mathrm{tg}^4(x) + a\ln^2(x) + a = 0$
8	$a\mathrm{tg}(x) + a\ln(\sin x) + a\ln(\mathrm{tg}(x)) + a^3\cos(x) + a = 0$
9	$\sqrt[3]{a/\cos(x)} + e^{-ax} + \mathrm{tg}(ax^2)\sin^2(ax) - a \cdot x^{3/4} + a = 0$
10	$ae^{-x} + \cos(ax) + \frac{1}{5x}\ln(ax) + ax^4 + a = 0$
11	$\ln\lvert ax\rvert + a\cos(x^4)\sin(x^{-3}) - a\ln^3(x) + \sqrt{x/a} + a = 0$

Окончание табл. 5.4

Номер варианта	Уравнение
12	$a\ln(xa)+\cos^2(ax)/\operatorname{tg}(ax)+\sqrt{(x-7)/a}+(x+7)^3a+a=0$
13	$a\sqrt{\ln(x)}+\sqrt{\operatorname{tg}(ax)}-ax^{4/5}+ax^3/\cos(x)+a=0$
14	$\ln\ (\sqrt{a}x)+x^3a^2+\sin(ax^4)-\sqrt{a^3}\operatorname{tg}(\sqrt{x})+a=0$
15	$ae^{-4x}+a\ln(x-5x^3)+a\sqrt{(x+a)^3}+$ $+\ln(a)/(x^3-2a)+a=0$
16	$\sqrt{a}\cos^3\lvert x\rvert+a\sin(ax)+a\operatorname{tg}^3(ax)+a\ln^2(ax)+a=0$
17	$a\cos^{3/4}(x/a)+a\sqrt{\sin(\ln a^2x)}+$ $+\ln(a\cdot\operatorname{tg}(x))+\cos(x^{a/2})+a=0$
18	$ax^5-\sqrt{\cos(a)}\sin\ (ax)+ae^{-ax}x+x^{-2a}+a=0$
19	$e^{-ax}+\cos(ax-ax^2)+\ln(\sqrt{\frac{ax}{x^3}})+\sin(ax^2)+a=0$
20	$ax^{3a}+a\ln^3(x)-\sqrt{xa}+a\ln(x^{-a})+a=0$
21	$\ln(ax)+\cos(x^a)/\ln(a^x)-(x+7a)+$ $+a\frac{e^{-2ax}}{\ln(ax^{2/7})}+a=0$
22	$a\sqrt{x^a}+\cos^{a/2}(x^3)+\ln(a\cdot\operatorname{tg}(x))\ +a\sqrt{\frac{e^{-x}}{x^a}}+a=0$
23	$\ln\ (\cos(ax))+a^2e^{-ax}-ax^5+\sqrt{a^3}\ln\lvert x\rvert+a=0$
24	$e^{-ax/3}+a\ln(ax)+a\sqrt{a^4\cos^2(x)}-a\ \ /(x^3+7a)+a=0$
25	$a\ln^{3/a}\lvert x\rvert+a\sin(ae^{-x})+(x-a)+a\frac{\operatorname{tg}^2(x^3)}{\ln(x-a)}+a=0$

Результатом решения задачи должны быть все решения уравнения $F(x, a) = 0$ в зависимости от параметра a. Пример оформления результатов решения уравнения $F(x, a) = 0$ представлен в табл. 5.5.

Таблица 5.5

Параметр a	Корни уравнения $F(x,a) = 0$	Число корней уравнения
a_1	x_{11}	1
a_2	x_{21}, x_{22}	2
a_3	x_{31}, x_{32}, x_{33}	3
a_4	x_{41}, x_{42}, x_{43}, x_{44}	4
a_5	x_{51}, x_{52}, x_{53}, x_{54}, x_{53}	5

Эта таблица должна иллюстрировать количество корней уравнения для $F(x, a) = 0$ (ограничиться пятью корнями при заданном параметре a). Построить график функции $F(x, a)$ от x для параметра a, соответствующего наибольшему количеству корней, найденных в задании. Если заданное число корней (5) не удалось найти, то следует объяснить почему.

5.6. **Задачи по разработке алгоритмов и программ для аппроксимации табулированных функций**

Постановка задачи. Разработать алгоритм и программу для выбора аппроксимирующего многочлена вида $y = A_0 + A_1x + A_2x^2 + ... + A_mx^n$ для табулированных функций y_1, y_2, …, y_6. Значения переменных x для построения многочлена $y_i = f(x)$ (исходные данные) представлены в табл. 5.6, варианты заданий – в табл. 5.7.

Таблица 5.6

Номер наблюдения	x	y_1	y_2	y_3	y_4	y_5	y_6
1	2	3	4	5	6	7	8
1	0,1	2,57	0,15	2,98	3,19	6,54	7,26
2	0,3	3,87	-0,01	3,45	1,59	6,63	7,95
3	0,5	5,67	0,46	3,84	0,4	6,83	8,77
4	0,7	8,39	1,71	3,87	0,22	7,34	9,72
5	0,9	12,9	3,94	3,24	1,95	8,44	10,78
6	1,1	20,47	7,4	1,73	6,84	10,48	12,45

Окончание табл. 5.6

1	2	3	4	5	6	7	8
7	1,3	32,84	12,41	-0,67	16,58	13,71	14,74
8	1,5	52,17	19,47	-3,78	33,4	18,7	18,18
9	1,7	81,1	29,21	-7,07	60,08	25,87	23,36
10	1,9	122,68	42,49	-9,62	100,1	35,76	30,99
11	2,1	180,44	60,43	-10,04	157,67	48,93	41,97
12	2,3	268,38	84,42	-6,39	237,83	65,97	57,38
13	2,5	360,93	116,18	3,89	346,5	87,52	78,45
14	2,7	493,01	157,8	24,08	490,6	114,23	106,65
15	2,9	660	211,78	58,21	678,09	146,8	143,64
16	3,1	867,76	281,04	111,19	918,06	185,93	191,28
17	3,3	1122,64	369	188,86	1220,08	232,35	251,71
18	3,5	1431,43	479,59	298,07	1597,9	286,78	327,27
19	3,7	1801,46	617,3	446,74	2062,29	349,99	420,58
20	3,9	2240,52	787,21	643,99	2628,34	422,73	534,52
21	4,1	2756,88	995,04	900,13	3311,95	505,74	672,24
22	4,3	3359,35	1247,17	1226,83	4130,59	599,78	837,2
23	4,5	4057,19	1550,71	1637,14	5103,4	705,6	1033,14
24	4,7	4860,21	1913,5	2145,58	6251,27	823,94	1264,13
25	4,9	5778,7	2344,18	2768,2	7596,91	955,52	1534,54
26	5,1	6823,5	2852,21	3532,71	9164,91	1101,05	1849,11
27	5,3	8005,85	3447,91	4428,49	10980,87	1261,21	2212,91
28	5,5	9337,7	4142,52	5506,72	13076,4	1436,67	2631,36
29	5,7	10831,38	4948,19	6780,41	15479,26	1628,06	3110,26
30	5,9	12499,79	5878,09	8274,53	18223,42	1835,97	3665,8
31	6,1	14356,37	6946,36	10016,04	21344,1	2060,97	4274,55
32	6,3	16415,09	8768,24	12033,99	24878,92	2303,59	4973,5
33	6,5	18690,45	9560,03	14359,59	28867,9	2564,29	5760,63
34	6,7	21197,51	11139,19	17026,29	33353,59	2843,52	6641,98
35	6,9	23951,86	12924,34	20069,86	38381,12	3141,65	7627,6
36	7,1	26969,66	14935,29	23528,45	43998,25	3459	8725,62
37	7,3	30267,6	17193,14	27442,7	50255,62	3795,83	9945,22
38	7,5	33862,96	19720,24	31855,77	57806,5	4152,36	11296,05
39	7,7	37773,55	22540,29	36813,46	64907,16	4528,7	12788,24
40	7,9	42017,77	25678,32	42364,27	73416,84	4924,93	14432,44

Таблица 5.7

Номер варианта	Вид функции	Степени аппроксимирующего многочлена
1	2	3
0	$y_1 = f(x)$	3, 5, 7
1	$y_2 = f(x)$	2, 4, 6
2	$y_3 = f(x)$	2, 3, 4

Окончание табл. 5.7

1	2	3
3	$y_4 = f(x)$	3, 4, 7
4	$y_5 = f(x)$	3, 4, 6
5	$y_6 = f(x)$	2, 3, 5
6	$y_2 = f(y_1)$	2, 7, 8
7	$y_3 = f(y_1)$	3, 5, 6
8	$y_4 = f(y_1)$	3, 7, 8
9	$y_5 = f(y_1)$	2, 5, 6
10	$y_6 = f(y_1)$	3, 4, 5
11	$y_3 = f(y_2)$	3, 4, 7
12	$y_4 = f(y_2)$	3, 6, 7
13	$y_5 = f(y_2)$	2, 3, 6
14	$y_6 = f(y_2)$	2, 4, 5
15	$y_4 = f(y_3)$	3, 4, 6
16	$y_5 = f(y_3)$	3, 5, 7
17	$y_6 = f(y_3)$	2, 4, 6
18	$y_5 = f(y_4)$	2, 3, 4
19	$y_6 = f(y_4)$	3, 4, 7
20	$y_6 = f(y_5)$	3, 5, 7
21	$\ln y_2 = f(x)$	2, 4, 6
22	$\ln y_3 = f(x)$	2, 3, 4
23	$\ln y_4 = f(x)$	3, 4, 7
24	$\ln y_5 = f(x)$	2, 3, 4
25	$\ln y_6 = f(x)$	3, 4, 7

При построении многочлена для функции $f(x) = \ln y$ необходимо данные $|y_i|$ линеаризовать, прологарифмировав данные столбца.

При разработке алгоритма и программы можно пользоваться программой для решения системы линейных уравнений (п. 3.1). При этом каждый студент решает свою систему линейных уравне-

ний, которую необходимо составить на основе данных табл. 5.6 и 5.5. Например, для расчета коэффициентов полинома 6-й степени $A_0, A_1, A_2, A_3, A_4, A_5, A_6$ необходимо решить систему уравнений вида

$$\begin{cases} t_0A_0 + t_1A_1 + t_2A_2 + t_3A_3 + t_4A_4 + t_5A_5 + t_6A_6 = C_0, \\ t_0A_0 + t_1A_1 + t_2A_2 + t_3A_3 + t_4A_4 + t_5A_5 + t_6A_6 = C_1, \\ t_0A_0 + t_1A_1 + t_2A_2 + t_3A_3 + t_4A_4 + t_5A_5 + t_6A_6 = C_2, \\ t_0A_0 + t_1A_1 + t_2A_2 + t_3A_3 + t_4A_4 + t_5A_5 + t_6A_6 = C_3, \\ t_0A_0 + t_1A_1 + t_2A_2 + t_3A_3 + t_4A_4 + t_5A_5 + t_6A_6 = C_4, \\ t_0A_0 + t_1A_1 + t_2A_2 + t_3A_3 + t_4A_4 + t_5A_5 + t_6A_6 = C_5, \end{cases}$$

где $t_0 = 40$, $t_1 = \sum_{i=1}^{40} x_i$, $t_2 = \sum_{i=1}^{40} {x_i}^2$, $t_3 = \sum_{i=1}^{40} {x_i}^3$, $t_4 = \sum_{i=1}^{40} {x_i}^4$, $t_5 = \sum_{i=1}^{40} {x_i}^5$, $t_6 = \sum_{i=1}^{40} {x_i}^6$, $C_0 = \sum_{i=1}^{40} y_i$, $C_1 = \sum_{i=1}^{40} x_i y_i$, $C_2 = \sum_{i=1}^{40} {x_i}^2 y_i$, $C_3 = \sum_{i=1}^{40} {x_i}^3 y_i$, $C_4 = \sum_{i=1}^{40} {x_i}^4 y_i$, $C_5 = \sum_{i=1}^{40} {x_i}^5 y_i$.

Результаты решения задачи должны включать найденные коэффициенты полиномов A_i и построение графика $y_i = f(x)$ вместе с исходными данными. Для трех различных полиномов (степень полинома, см. табл. 5.6) рассчитать среднеквадратическое отклонение

$$\delta = \sqrt{\frac{1}{n}\sum_{i=1}^{40} (y(x) - y_i)^2},$$

где y_i – табулированное значение;

$y(x)$ – аппроксимированный многочлен, по которому нужно оценить близость полинома к исходным данным (см. табл. 5.5).

Сделать выводы о лучшей степени аппроксимации функции.

Таким образом, студент, выполняющий 4-й вариант, должен построить полиномы вида

$$y_5 = A_0 + A_1 x + A_2 x^2 + A_3 x^3,$$
$$y_5 = A_0 + A_1 x + A_2 x^2 + A_3 x^3 + A_4 x^4,$$
$$y_5 = A_0 + A_1 x + A_2 x^2 + A_3 x^3 + A_4 x^4 + A_5 x^5 + A_6 x^6$$

и вычислить среднеквадратическое отклонение для каждого полинома.

5.7. **Задачи по разработке алгоритмов и программдля численного интегрирования и дифференцирования функции**

Постановка задачи. Разработать алгоритм и программу для численного дифференцирования и интегрирования функции $f_i(x)$. Варианты функции представлены в табл. 5.1 (a = 1, b = 1, c = 1, d = 1). Пределы интегрирования функции равны $x_H = 2$, $x_K = 10$. При построении программы воспользоваться формулами прил. 2. Студенты, выполняющие четный вариант, при выполнении интегрирования реализуют метод Симпсона, нечетный вариант – метод трапеций. По разработанной программе рассчитать значение интеграла для различных шагов разбиения. Построить зависимость или таблицу значений интеграла от шагов разбиения.

Таким образом, студент, выполняющий вариант 4, должен вычислить дифференциальную кривую $\frac{df}{dx}$, воспользовавшись формулой

$$\frac{df(x)}{dx} = \frac{f(x+\Delta x) - f(x)}{\Delta x},$$

где Δx – шаг аргумента (выбирается самостоятельно, например Δx_1 = 0,1; Δx_2 = 0,01; Δx_3 = 0,001).

Так, для 4-го варианта $f(x) = \sqrt{x} + \sqrt{\ln(x)} + x^3 + x^{1/10}$. Пример оформления результатов дифференцирования представлен в табл. 5.8.

Таблица 5.8

i	$x_2 - x_1$	$x_3 - x_2$	$x_4 - x_3$	…	$x_n - x_{n-1}$
Δf_i	$f(x_2) - f(x_1)$			…	
$\frac{df_i}{dx}$	$(f(x_2) - f(x_1))/(x_2 - x_1)$			…	

Написать алгоритм и программу вычисления интеграла

$$S = \int_2^{10} [\sqrt{x} + \sqrt{\ln x} + x^3 + x^{1/10}]dx,$$

воспользовавшись формулой Симпсона (4 – четный вариант). Построить график зависимости S (значение интеграла) = $f(\Delta x_i)$ (Δx_i – шаг разбиения). Сделать выводы. Предложить варианты расчета интеграла с заданной точностью $\varepsilon = 10^{-6}$. В отчет необходимо включить следующие результаты: дифференциальная кривая, значение интеграла для разных шагов разбиения.

5.8. **Задачи по разработке алгоритмов и программ для численного решения дифференциальных уравнений**

Постановка задачи. Разработать алгоритм и программу для решения дифференциального уравнения вида $y' = f(x, y)$ методом Рунге-Кутта. Варианты функций $f(x, y)$ представлены в табл. 5.1 и должны быть сформулированы по правилу $a = b = c = d = y$.

Граничные условия выбрать с учетом формул

$$x_0 = \frac{n+1}{25},\ x_n = \frac{n+1}{25} + 2,\ y_0 = \frac{n+4}{25},\ k = \begin{cases} 3n + 20, & n \le 10, \\ 3n - 5, & n > 10, \end{cases}$$

где n – вариант заданий.

Уравнение решается на отрезке $[x_0, x_n]$ при разбиении его на k (шаг) равных частей . Точность вычислений $\varepsilon = 10^{-5}$.

Исследовать влияние числа разбиений k на вид решения $y_i(x)$.

Таким образом, студенты, решающие 4-й вариант, должны написать программу по решению уравнения

$$y' = y\sqrt{x} + \sqrt{\ln(yx)} + yx^3 + yx^{1/10} + y,$$

на участке $x_0 = \frac{4+1}{25} = 0{,}2$, $x_n = \frac{4+1}{25} + 2 = 2{,}2$ для начального значения $y_0 = \frac{4+4}{25}$ при $k = 3 \cdot 4 + 20 = 32$.

В отчете необходимо представить постановку задачи, алгоритм, программу и результаты, содержащие табулированную функцию $y = f(x)$ для трех различных шагов разбиения. Для выбора первого шага (k) воспользоваться вышеприведенной формулой. Два других шага выбрать самостоятельно. Построить графики $y = f(x)$ для участка $[x_0, x_n]$ для различных шагов разбиения.

5.9. **Задачи по разработке технологии численного решения уравнений в частных производных в MS EXCEL**

Постановка задачи. Разработать алгоритм и программу для решения уравнения

$$\frac{\partial T}{\partial t} = q\left(\frac{\partial T^2}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right),$$

где $T(t, x, y)$ – температура в момент t для координат области x, y;

q – коэффициент температуропроводности.

Вид двухмерной плоскости задается на основе рис. 5.1.

В прямоугольной области среды располагаются две отливки прямоугольной формы из разных металлов. Размеры прямоугольников, области расположения отливок, начальные значения температуры клеток областей (отливок и среды), коэффициент температуропроводности выбираются по данным, представленным в табл. 5.9. В табл. 5.9 значения a и b – это высота и ширина области среды, a_1

и b_1 – это высота и ширина отливки *1*, a_2 и b_2 – это высота и ширина отливки *2* (размерность значений – число клеток в MS EXCEL), q_1, q_2, q – коэффициенты температуропроводности соответственно отливки *1*, отливки *2*, области среды. Начальная температура отливки *1* вычисляется по формуле

$$T_1 = 300 + 50 \cdot n;$$

начальная температура отливки *2*

$$T_2 = 700 + n^2;$$

начальная температура области среды

$$T = 10 + n,$$

где n – номер варианта.

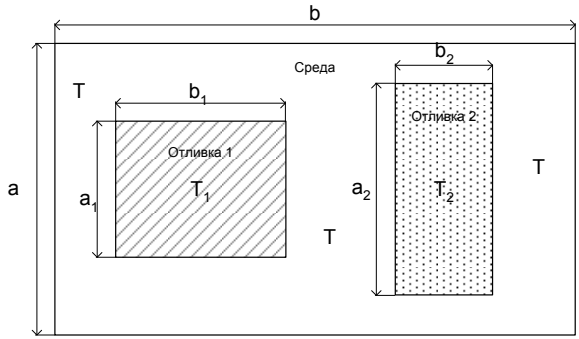


Рис. 5.1

Рассчитать температурное поле охлаждения отливок для трех случаев:

– для заданных коэффициентов температуропроводности (q_1, q_2, q_3);

– коэффициенты температуропроводности повышены на 50 %;

– коэффициенты температуропроводности повышены на 75 %.

Таблица 5.9

Номер варианта, n	a_1	b_1	a_2	b_2	a	b	q_1	q_2	q
1	8	25	16	10	20	50	30	45	70
2	10	30	18	8	21	49	31	44	69
3	10	40	17	6	22	48	32	43	68
4	12	32	20	12	23	47	33	42	67
5	14	33	19	10	24	46	34	41	66
6	10	35	21	14	25	45	35	40	65
7	15	37	23	15	26	44	36	39	64
8	8	38	24	13	27	43	37	38	63
9	12	39	26	12	28	42	38	37	62
10	10	25	25	11	29	40	39	3	61
11	11	26	28	10	30	40	40	34	60
12	7	27	15	8	20	41	41	35	59
13	10	28	16	7	21	42	42	33	58
14	12	29	19	10	22	43	43	32	57
15	13	30	21	9	23	44	44	31	56
16	14	25	20	8	24	45	45	30	55
17	12	26	23	6	25	46	46	50	54
18	8	27	17	6	26	47	47	49	53
19	9	28	25	10	27	48	48	48	52
20	10	22	26	13	28	49	49	47	51
21	14	19	25	10	29	50	50	46	50
22	12	25	27	8	30	49	51	45	70
23	7	15	13	6	20	48	52	44	69
24	6	30	14	5	21	47	53	43	68
25	10	22	16	7	22	46	54	42	67

Так, например, для варианта 2 ($n = 2$) условия задачи будут задаваться следующим образом.

Шаг 1. Определение размеров прямоугольников и коэффициентов температуропроводности из табл. 5.9:

$a_1 = 10$ клеток, $b_1 = 30$ клеток, $a_2 = 18$ клеток, $b_2 = 8$ клеток,

$a = 21$ клетка, $b = 49$ клеток,

$q_1 = 31$, $q_2 = 44$, $q = 69$.

Шаг 2. Определение начальных температур отливок и области среды по формулам

$$T_1 = 300 + 50 \cdot 2 = 400\ ^\circ\text{C},$$

$$T_2 = 700 + n^2 = 700 + 2^2 = 704\,^\circ\text{C},$$

$$T = 10 + 2 = 12\,^{\circ}\text{C}.$$

Шаг 3. Построение в MS EXCEL клеточной структуры среды и отливки. Для того чтобы среда отображалась во весь экран без прокрутки, необходимо уменьшить ширину столбцов, для этого нужно выделить столбцы, которые необходимо уменьшить, выбрать *Формат – Столбец – Ширина* и в появившемся диалоговом окне задать ширину столбцов, например, 2. Для корректного отображения температуры в клетках необходимо уменьшить шрифт. Вид полученной клеточной структуры представлен на рис. 5.2.

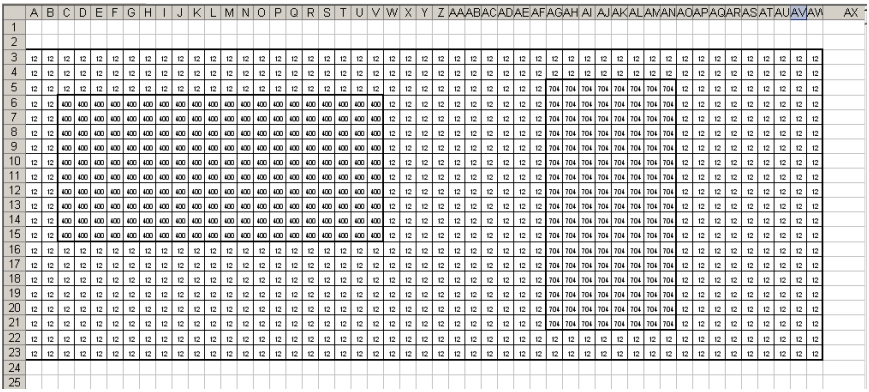


Рис. 5.2

5.10. **Задачи по разработке технологии решения системы линейных уравнений в** MS EXCEL

Постановка задачи. Разработать алгоритм решения системы линейных уравнений с пятью неизвестными в MS EXCEL, если коэффициенты системы вычисляются по закону, заданному в табл. 5.10. Например, для 1-го варианта коэффициенты вычисляются как

$$a_{ij} = \sin(i^2 + 2ij - j),\ c_i = \text{tg}^3(i + 10),$$

где i = 1; 5 (1), j = 1; 5 (1), (1 – шаг изменения переменной).

Система линейных уравнений после вычисления коэффициентов должна иметь вид

$$\begin{cases} a_{11}^2 x_1 + a_{12} x_2 + a_{13} x_3 + a_{14} x_4 + a_{15} x_5 = c_1, \\ a_{21} x_1 + a_{22}^2 x_2 + a_{23} x_3 + a_{24} x_4 + a_{25} x_5 = c_2, \\ a_{31} x_1 + a_{32} x_2 + a_{33}^2 x_3 + a_{34} x_4 + a_{35} x_5 = c_3, \\ a_{41} x_1 + a_{42} x_2 + a_{43} x_3 + a_{44}^2 x_4 + a_{45} x_5 = c_4, \\ a_{51} x_1 + a_{52} x_2 + a_{53} x_3 + a_{54} x_4 + a_{55}^2 x_5 = c_5. \end{cases}$$

Решить сформированную систему линейных уравнений, использовав встроенные функции MS EXCEL, предварительно определив коэффициенты при неизвестных j по формулам, представленным в табл. 5.10 по вариантам заданий.

Таблица 5.10

Номер варианта	Число неизвестных j	a_{ij}	c_i
1	2	3	4
1	8	$\sin(i^2+2ij-j)$	$\mathrm{tg}^3(i+10)$
2	9	$\cos(j^2-3ij+i^2)$	$3\sin(i+5)$
3	10	$\ln(i+j)$	$2\sin(i^2-3)$
4	11	$\mathrm{tg}(i+j+10)$	i^2+2i-3
5	12	$\log_2(i+j^2)$	$2\sin^3(5i^2)$
6	8	$\sin^3(i+12ij-j)$	$\cos(i^3+i^2)$
7	9	$\mathrm{tg}^2(i^2+j)$	i^4+2i
8	10	$\log_{12}^2(i+j^2)$	$12\sin^2(i^2)$
9	11	$13\sin^3(i+5j)$	$\ln(10i^2+3i)$
10	12	$5\sin(i+5j)$	$\mathrm{ctg}^3(i+1)$
11	8	$\log_{12}(2i+j^2)$	$\cos(2i^3+i^2)$
12	9	$\sin(i^2+8ij-j)$	$\mathrm{tg}^2(6i+10)$
13	10	$3\cos(j^2-6i+2i^2)$	$3\sin(i+5)$

Продолжение табл. 5.10

1	2	3	4
14	11	$10\ln(i+j)$	$\sin(3i^2-3)$
15	12	$4\mathrm{tg}^2(i+4j+8)$	i^2+23
16	8	$8\log_5(5i+j^2)$	$-5\mathrm{tg}(2i^2+4j)$
17	9	$\sin^2(i+22ij-j)$	$14\cos(i^3+4i^2)$
18	10	$8\mathrm{tg}^2(i^2+8j)$	i^4-52i
19	11	$\log_{12}^2(i+j^2)$	$25\sin^2(i^5)$
20	12	$13\sin^3(i+5j)$	$7\ln(10i^2+3i)$
21	8	$5\sin(2i+5j)$	$2\mathrm{ctg}^3(i+7)$
22	9	$2\log_2^2(2i+j^2)$	$3\cos^2(2i^3+3i^2)$
23	10	$3\sin(3i^2+3ij-j)$	$2\mathrm{tg}^2(i+2)$
24	11	$2\cos(i^3-3ij^2)$	$1{,}5\sin(i+1{,}5)$
25	12	$4\ln(i+4j)$	$5\sin(i^5-5)$
26	8	$2\mathrm{tg}(2i+j)$	$4i^2+4i-14$

Продолжение табл. 5.10

1	2	3	4
27	9	$-1{,}8\operatorname{tg}(2i^2+8j)$	$-5\log_3(i^2+3)$
28	10	$2\sin^2(2i+1{,}5ij-j)$	$2\cos^3(i^3+3i^2)$
29	11	$7\operatorname{tg}^2(i^2+7j)$	$14i^6+6i$
30	12	$7\log_7^2(7i+j^2)$	$4\sin^4(4i^2)$
31	8	$8\sin^2(8i+5j)$	$3\ln(i^2+3)$
32	9	$6\sin(i+4j)$	$\operatorname{tg}^3(i^3+1)$
33	10	$3\log_3^2(3i+j^2)$	$4\cos(4i^3+i)$
34	11	$8\sin(3i^2+2ij)$	$5\operatorname{tg}^3(i+15)$
35	12	$1{,}5\cos^2(1{,}5j^2+i^2)$	$4\sin^2(5i+15)$
36	8	$7\ln(i+7j)$	$2\sin^2(i^2-2)$
37	9	$3\operatorname{tg}^2(i+3j)$	$4i^2+2i-18$
38	10	$2\sin^5(5i^2)$	$-4\log_2^2(4i^2+4)$
39	11	$3\sin^3(i+ij-33j)$	$-5\operatorname{tg}(2i^2+5)$

Окончание табл. 5.10

1	2	3	4
40	12	$-1{,}5\operatorname{tg}(2i^2 + j)$	$7i^2 + 5i$
41	8	$5\log_5(5i + j^2)$	$2{,}5\sin^2(5i^2)$
42	9	$13\sin^3(i + 5j)$	$4\ln(4i^2 + 3i)$
43	10	$3\sin^2(2i + 3j)$	$\operatorname{tg}^2(5i + 1)$
44	11	$3\log_8(2i + 3j^2)$	$4\cos(4i^3 + 3i^2)$
45	12	$3\sin^3(i^2 - j)$	$4\operatorname{tg}^2(i + 12)$
46	8	$7\cos(7j^2 - 7ij + i^2)$	$3\sin^2(2i + 5)$
47	9	$7\ln(i + j)$	$5\sin^2(i^2 - 8)$
48	10	$2\operatorname{tg}^2(i + 2j + 12)$	$5i^2 + 2i - 5$
49	11	$5\log_5(i + 5j^2)$	$4\operatorname{tg}^2(4i + 2)$
50	12	$4\sin(i + 12ij - 4j)$	$\cos^2(i^3 + 2i^2)$

5.11. **Задачи по разработке технологии решения нелинейных уравнений в** MS EXCEL

Постановка задачи. Разработать алгоритм решения нелинейных уравнений вида $f(x) = 0$, используя метод дихотомии на отрезке $[a, b]$. Определить число корней на указанном отрезке, используя графические средства MS EXCEL. Найти три любых корня уравнения с точностью $\varepsilon = 0{,}01$. Варианты заданий представлены в табл. 5.11.

Таблица 5.11

Номер варианта	Функция $f(x)$	a	b
1	2	3	4
1	$\sin 4\left(x+\pi/4\right)$	-10	10
2	$5\cos\left(x-\pi/4\right)+\sin\left(x\right)$	-5	15
3	$4\cos\left(\frac{1}{x+1}\right)+\cos\left(x\right)$	-11	21
4	$\sin\left[\cos^2\left(x+4\pi\right)\right]$	-14	6
5	$\cos\left(3x\right)+2\sin\left(6x\right)+\cos\left(x\right)$	-5	25
6	$\mathrm{tg}\left(4x\right)+\mathrm{ctg}\left(x\right)$	-16	8
7	$4\cos\left(x\right)+x^2+2x$	-13	40
8	$5\sin\left(1x\right)+2\sin^2\left(2x\right)+3$	-8	30
9	$\log_4\left(x+12\pi\right)+\log_5\left(x+13\pi\right)$	-17	8
10	$\cos^3\left(3x\right)+2\cos^2\left(3x\right)$	-20	14
11	$\sin(3x)+\sin(6x)$	-12	18
12	$\mathrm{tg}(5x)+\mathrm{tg}(7x)$	-19	5
13	$\mathrm{ctg}(x)+\mathrm{ctg}(2x)$	-4	24
14	$\cos^3\left(3x\right)+2\cos^2\left(3x\right)$	-5	17
15	$\sin^4\left(3x\right)+4\sin^2\left(3x\right)$	-10	10
16	$\mathrm{tg}^2\left(3x\right)+\mathrm{tg}\left(4x\right)$	-5	15

Продолжение табл. 5.11

1	2	3	4
17	$\operatorname{ctg}(4x)+\operatorname{ctg}^3(12x)$	-11	21
18	$\sin^2(4x)+\cos^3(x)+\sin(4x+\pi)$	-14	6
19	$\sin 4\left(x+\pi/4\right)$	-5	25
20	$5\cos\left(x-\pi/4\right)+\sin(x)$	-16	8
21	$3\operatorname{tg}\left(x+\pi/10\right)+4\cos\left(\frac{1}{x+1}\right)+\cos(4x)$	-13	40
22	$\sin\left[\cos^2(x+4\pi)\right]+\ln(x+8)$	-8	30
23	$\cos(3x)+2\sin(3x)+\cos(4x)$	-17	8
24	$\operatorname{tg}(4x)+\operatorname{ctg}(x)+\operatorname{tg}(12x)$	-20	14
25	$4\cos(x)+x^2+2x$	-12	18
26	$5\sin(12x)+2\sin^2(2x)+3$	-19	5
27	$\log_5(x+13\pi)+\log_6(x+14\pi)$	-4	24
28	$\cos^3(3x)+3\cos(3x)$	-5	17
29	$\sin(3x)+\sin(6x)+\sin(9x)+\sin(11x)$	-10	10
30	$\operatorname{tg}(x)+\operatorname{tg}(3x)+\operatorname{tg}(5x)+\operatorname{tg}(7x)$	-5	15
31	$\operatorname{ctg}(x)+\operatorname{ctg}(2x)+\operatorname{ctg}(7x)+\operatorname{ctg}(9x)$	-11	21
32	$\cos^3(3x)+2\cos^2(3x)+3\cos(3x)+7$	-14	6
33	$\sin^4(3x)+4\sin^2(3x)+77$	-5	25
34	$\operatorname{tg}^2(3x)+\operatorname{tg}(4x)+2\operatorname{ctg}(x)$	-16	8
35	$\operatorname{ctg}(4x)+\operatorname{ctg}^3(12x)+\operatorname{ctg}^2(3x)$	-13	40
36	$\sin^2(4x)+\cos^3(x)+\sin(4x+\pi)$	-8	30
37	$\ln(x+5\pi)+\sin 4\left(x+\pi/4\right)+80$	-17	8
38	$\log_2(x+10\pi)+5\cos\left(x-\pi/4\right)+\sin(x)$	-20	14

Окончание табл. 5.11

1	2	3	4
39	$3\operatorname{tg}(x+\pi/10)+4\cos\left(\frac{1}{x+1}\right)+\cos(x)$	-12	18
40	$\sin\left[\cos^2(x+4\pi)\right]+\ln(x+8)$	-19	5
41	$\cos(x)+2\sin(6x)+\cos(x)$	-4	24
42	$\operatorname{tg}(x)+\operatorname{ctg}(x)+\operatorname{tg}(1x)$	-14	6
43	$4\cos(x)+x^2+2x$	-5	25
44	$5\sin(1x)+2\sin^2(2x)+3$	-16	8
45	$\log_4(x+12\pi)+\log_5(x+13\pi)+\log_6(x+14\pi)$	-13	40
46	$\cos^3(x)+2\cos^2(x)+3\cos(x)+7$	-8	30
47	$\sin(3x)+\sin(6x)+\sin(9x)+\sin(11x)$	-17	8
48	$\operatorname{tg}(x)+\operatorname{tg}(3x)+\operatorname{tg}(5x)+\operatorname{tg}(7x)$	-20	14
49	$\operatorname{ctg}(x)+\operatorname{ctg}(2x)+\operatorname{ctg}(7x)+\operatorname{ctg}(9x)$	-12	18
50	$\cos^3(x)+2\cos^2(x)+3\cos(x)+7$	-19	5

5.12. **Задачи по разработке технологии аппроксимации полиномов табулированных функций в** MS EXCEL

Постановка задачи. Построить два аппроксимационных полинома с различными степенями по табличным функциям, определенным в задании. Значения функций заданы в табл. 5.12.

Таблица 5.12

Номер варианта	x	y_1	y_2	y_3	y_4	y_5	y_6	y_7
1	2	3	4	5	6	7	8	9
1	0,1	1,345	-8,003	-19,0	-8	9,99	69,9	-170
2	0,6	1,6	1,352	9,0	-8,0	9,99	69,998	-170,2
3	1,1	6,005	8,007	29,0	-8,4	9,19	69,2	-170,3
4	1,6	35,56	9,712	34,1	-8,1	9,76	69,5	-170,5
5	2,1	126,265	4,217	17,6	-8,2	9,373	69,11	-171,06
6	2,6	329,12	-10,728	-27,1	-8,5	9,94	69,39	-176,08
7	3,1	70,125	-37,373	-107,1	-8,8	9,91	69,56	-179,9
8	3,6	1350,28	-77,968	-228,8	-8,13	9,866	69,925	-170,011

Продолжение табл. 5.12

1	2	3	4	5	6	7	8	9
9	4,1	235,585	-134,763	-399,2	-8,19	9,798	69,929	-170,012
10	4,6	387,04	-210,008	-624,9	-8,27	9,711	70,068	-170,013
11	5,1	580,645	-305,953	-912,7	-8,365	9,6079	70,	-170,014
12	5,6	8647,4	-424,848	-1269,3	-8,47	9,482	70,027	-170,015
13	6,1	1223,31	-568,943	-1701,51	-8,59	9,3304	70,06	-170,015
14	6,6	179,36	-740,488	-2215,92	-8,73	9,1522	70,143	-170,014
15	7,1	23041,57	-941,733	-2819,31	-8,891	8,9466	70,15	-170,012
16	7,6	3000,09	-1174,93	-3518,4	-9,06	8,7092	70,235	-170,008
17	8,1	353,43	-1442,32	-4319,89	-9,245	8,4404	70,328	-170,001
18	8,6	500,08	-1746,17	-5230,45	-9,437	8,193	70,443	-169,991
19	9,1	6376,89	-2088,71	-6256,78	-9,63	7,7941	70,593	-169,976
20	9,6	79204,84	-2472,21	-7405,51	-9,834	7,41374	70,78657	-169,954
21	10,1	97409,95	-2898,9	-8683,31	-10,09	6,992345	71,02071	-169,923
22	10,6	118583,2	-3371,05	-10096,8	-10,28	6,52795	71,30643	-169,881
23	11,1	143030,6	-3890,89	-11652,5	-10,478	6,018553	71,65176	-169,826
24	11,6	171073,2	-4460,69	-13357,2	-10,53	5,462163	72,06554	-169,752
25	12,1	203046,9	-5082,68	-15217,2	-10,09	4,856804	72,55744	-169,657
26	12,6	2902,7	-559,13	-1739,2	-10,253	4,251	73,18	-169,534
27	13,1	2006,7	-6492,27	-19429,7	-10,83	3,491327	73,815	-169,377
28	13,6	3239,9	-7284,37	-2795	-10,88	2,727314	74,6179	-169,18
29	14,1	3798,2	-8137,66	-241,8	-10,866	1,906	75,53078	-168,934
30	14,6	4392,6	-9054,41	-276,2	-10,76	1,027	76,59003	-168,63
31	15,1	4949,2	-10036,9	-3004,8	-10,4786	0,0875	77,80497	-168,255
32	15,6	56309	-11087,2	-333,8	-10,13	-0,916	79,19216	-167,798
33	16,1	64567,9	-12207,8	-3669,5	-9,70228	-1,982	80,766	-167,244
34	16,6	73077	-13400,9	-4008,1	-9,11796	-3,112	82,555	-166,576
35	17,1	8232,2	-14668,6	-4375,7	-8,37	-4,31	84,569	-165,776
36	17,6	9254,5	-16013,3	-4778,6	-7,467	-5,582	86,8362	-164,822
37	18,1	106140	-17437,2	-5202,6	-6,36	-6,93	89,312	-163,69
38	18,6	11620	-18942,6	-56463,9	-5,04	-8,346	92,223	-162,354
39	19,1	1287309	-20531,6	-6118,3	-3,482	-9,832	95,319	-160,784
40	19,6	14270	-22206,6	-66121,7	-1,66	-11,48	98,867	-158,947
41	20,1	15579	-23969,8	-712,9	0,4467	-13,56	102,777	-156,806
42	20,6	1465	-25823,4	-76798,5	2,8712	-14,748	107,35	-154,319
43	21,1	1915	-27769,8	-8253,2	5,6427	-16,51	111,794	-151,443
44	21,6	2111	-29811,1	-88539,6	8,781	-18,47	116,937	-148,128
45	22,1	237	-31949,6	-94823	12,31	-20,42	122,52	-144,318
46	22,6	2536	-34187,5	-10139	16,34	-22,5425	128,891	-139,954
47	23,1	2863	-36527,2	-108242	20,78	-24,73	135,706	-134,971
48	23,6	3089	-38970,8	-1159	25,778	-26,91	143,134	-129,297
49	24,1	3211	-41520,6	-122833	31,34	-29,2938	151,227	-122,853
50	24,6	3538	-44178,8	-130580	37,4297	-31,711	160,016	-115,555
51	25,1	3898	-46947,8	-138634	44,181	-34,265	169,5498	-107,311
52	25,6	41931	-49829,6	-14701	51,61	-36,8957	179,8744	-98,0192
53	26,1	45493	-5226,7	-15685	59,74	-39,6264	191,038	-87,5727
54	26,6	48055	-5541,3	-16689	68,26	-42,4582	203,0907	-75,8536

Окончание табл. 5.12

1	2	3	4	5	6	7	8	9
55	27,1	52603	-59175,5	-17420	78,45	-45,3927	216,0847	-62,7355
56	27,6	562938	-6231,7	-13680	89,08308	-48,4315	230,074	-48,0817
57	28,1	6097877	-66012,1	-193674	100,6349	-51,5763	245,1147	-31,7454
58	28,6	65249	-69619	-204005	113,16	-54,8285	261,2649	-13,5685
59	29,1	70102	-73354,5	-21478	126,745	-58,1898	278,5851	6,618548
60	29,6	7516696	-77221	-2295	141,423	-61,6618	297,1375	28,99778
61	30,1	80407	-81220,7	-23060	157,264	-65,2459	316,9868	53,76401
62	30,6	8591	-85355,8	-24777	174,338	-68,9437	338,1998	81,12576
63	31,1	91690	-89628,7	-26848	192,69	-72,7567	360,8457	111,3058
64	31,6	97778	-9041,5	-273276	212,42	-76,6865	384,9958	144,542
65	32,1	104168	-9896,5	-26063	233,59	-80,7345	410,7238	181,0881
66	32,6	110830	-103296	-29213	256,26	-84,9023	438,1058	221,2142
67	33,1	117880	-108142	-31727	280,55	-89,1912	467,2203	265,208
68	33,6	125129	-113137	-326606	306,4	-93,6028	498,12	313,3752
69	34,1	13773	-118283	-340854	334,9	-98,1384	530,9	366,07
70	34,6	1474	-123583	-355472	363,92	-102,8	565,78	423,92
71	35,1	1496	-129039	-370460	395,42	-107,588	602,65	486,64
72	35,6	1572	-1352	-385820	428,36	-112,504	641,70	554,794
73	36,1	166917	-140426	-401553	463,9	-117,551	682,99	628,84
74	36,6	1762	-1462	-417658	501,4	-122,72	726,64	709,61
75	37,1	181923	-152462	-434138	541,67	-128,08	772,74	797,38
76	37,6	196312	-15830	-450990	584,56	-133,42	821,40	892,39
77	38,1	207373	-165167	-468216	628,9	-139,02	872,71	995,22
78	38,6	21439	-1775	-485814	676,32	-144,778	926,79	1106,65
79	39,1	230055	-178557	-503784	726,53	-150,61	983,75	1227,4
80	39,6	2410982	-1815	-522125	779,51	-156,24	1043,69	1356,977

Варианты заданий представлены в табл. 5.13.

Таблица 5.13

Номер варианта	Виды функции	Максимальные степени многочленов
1	2	3
1	$y_1 = f(x)$, $y_2 = f(y_1)$	3, 4, 5
2	$y_2 = f(x)$, $y_3 = f(y_1)$	4, 5, 6
3	$y_3 = f(x)$, $y_4 = f(y_1)$	3, 5, 7
4	$y_4 = f(x)$, $y_5 = f(y_1)$	4, 6, 8
5	$y_5 = f(x)$, $y_6 = f(y_1)$	2, 4, 6
6	$y_6 = f(x)$, $y_7 = f(y_1)$	5, 6, 7
7	$y_7 = f(x)$, $y_3 = f(y_2)$	4, 5, 8
8	$y_1 = f(y_3)$, $y_2 = f(y_4)$	3, 4, 5
9	$y_2 = f(y_3)$, $y_3 = f(y_4)$	4, 5, 6

Продолжение табл. 5.13

1	2	3
10	$y_3 = f(y_5)$, $y_4 = f(y_6)$	3, 5, 7
11	$y_4 = f(y_3)$, $y_5 = f(y_4)$	4, 6, 8
12	$y_5 = f(y_3)$, $y_6 = f(y_4)$	2, 4, 6
13	$y_6 = f(y_3)$, $y_7 = f(y_4)$	5, 6, 7
14	$y_7 = f(y_3)$, $y_3 = f(y_4)$	4, 5, 8
15	$y_1 = f(y_5)$, $\ln(y_2) = f(y_1)$	3, 4, 5
16	$y_2 = f(y_5)$, $\ln(y_3) = f(y_1)$	4, 5, 6
17	$y_3 = f(y_5)$, $\ln(y_4) = f(y_1)$	3, 5, 7
18	$y_4 = f(y_5)$, $\ln(y_5) = f(y_1)$	4, 6, 8
19	$y_5 = f(y_6)$, $\ln(y_6) = f(y_1)$	2, 4, 6
20	$y_6 = f(y_5)$, $\ln(y_7) = f(y_1)$	5, 6, 7
21	$y_7 = f(y_5)$, $\ln(y_3) = f(y_2)$	4, 5, 8
22	$y_1 = f(y_6)$, $\ln(y_2) = f(y_4)$	3, 4, 5
23	$y_2 = f(y_6)$, $\ln(y_3) = f(y_4)$	4, 5, 6
24	$y_3 = f(y_6)$, $\ln(y_4) = f(y_6)$	3, 5, 7
25	$y_4 = f(y_6)$, $\ln(y_5) = f(y_4)$	4, 6, 8
26	$y_5 = f(y_7)$, $\ln(y_6) = f(y_4)$	2, 4, 6
27	$y_6 = f(y_6)$, $\ln(y_7) = f(y_4)$	5, 6, 7
28	$y_7 = f(y_6)$, $\ln(y_3) = f(y_4)$	4, 5, 8
29	$\ln(y_1) = f(x)$, $y_2 = f(y_1)$	3, 4, 5
30	$\ln(y_2) = f(x)$, $y_3 = f(y_2)$	4, 5, 6
31	$\ln(y_3) = f(x)$, $y_4 = f(y_2)$	3, 5, 7
32	$\ln(y_4) = f(x)$, $y_5 = f(y_2)$	4, 6, 8
33	$\ln(y_5) = f(x)$, $y_6 = f(y_2)$	2, 4, 6
34	$\ln(y_6) = f(x)$, $y_7 = f(y_2)$	5, 6, 7
35	$\ln(y_7) = f(x)$, $y_3 = f(y_2)$	4, 5, 8
36	$y_1 = f(x)$, $y_2 = f(y_1)$	4, 5, 6
37	$y_2 = f(x)$, $y_3 = f(y_1)$	5, 6, 7
38	$y_3 = f(x)$, $y_4 = f(y_1)$	6, 7, 8
39	$y_4 = f(x)$, $y_5 = f(y_1)$	3, 4, 5
40	$y_5 = f(x)$, $y_6 = f(y_1)$	4, 5, 6
41	$y_6 = f(x)$, $y_7 = f(y_1)$	5, 6, 7
42	$y_7 = f(x)$, $\ln(y_3) = f(y_2)$	4, 5, 8
43	$y_1 = f(y_3)$, $\ln(y_2) = f(y_1)$	3, 4, 5
44	$y_2 = f(y_3)$, $\ln(y_3) = f(y_4)$	4, 5, 6
45	$y_3 = f(y_5)$, $y_4 = f(y_6)$	3, 4, 5
46	$y_4 = f(y_3)$, $y_5 = f(y_4)$	4, 5, 6

Окончание табл. 5.13

1	2	3
47	$y_5 = f(y_3)$, $y_6 = f(y_4)$	5, 6, 7
48	$y_6 = f(y_3)$, $y_7 = f(y_4)$	7, 8, 9
49	$y_7 = f(y_3)$, $y_3 = f(y_4)$	3, 5, 7
50	$y_1 = f(y_5)$, $\ln(y_2) = f(y_1)$	4, 6, 8

Если вид функции дается с использованием логарифма, то соответствующий столбец необходимо прологарифмировать. Например, при построении многочлена для функции $\ln(y_7) = f(x)$ необходимо столбец y_7 прологарифмировать, а многочлен должен быть построен по новым значениям столбца.

5.13. **Задачи по разработке технологии численного дифференцирования функций в** MS EXCEL

Постановка задачи. Провести табуляцию функции, заданной в табл. 5.14, в соответствии с вариантом задания на интервале $[-3\pi, 3\pi]$. Шаг табуляции выбрать самостоятельно. Численно продифференцировать эту функцию, определив $f'(x)$, $f''(x)$, $f'''(x)$, $f''''(x)$. Использовать формулы для левых, правых и центральных разностей. Графически сравнить результаты численного дифференцирования, изобразив на одном рисунке дифференциальную кривую, полученную тремя методами. Вычислить суммы, составленные из дифференциальных кривых для трех шагов табуляции: Δx – первичный шаг табуляции; $\Delta x_1 = \Delta x/2$; $\Delta x_2 = \Delta x_2/2$; $\Delta x_3 = \Delta x/2$;

$$S_1 = \sum_{i=1}^{n_1} f'(x_i); \quad S_2 = \sum_{i=1}^{n_1} \sin f'(x_i); \quad S_3 = \sum_{i=1}^{n_2} f''(x)_i; \quad S_4 = \sum_{i=1}^{n_2} f''(x_i) + ;$$

$$+ \sum_{i=1}^{n_2} \operatorname{tg} f''(x_i) \; S_5 = \sum_{i=1}^{n_3} f'''(x_i); \; S_5 = \sum_{i=1}^{n_3} \left[f'''(x_i) + \left[f'''(x_i) \right]^2 \right].$$

Провести процесс сглаживания по заданному количеству точек для функций $f''(x)$, $f'''(x)$. Построить графики исходной и сглаженной кривой. Для сглаживания функций использовать усреднения соседних точек по формуле

$$yy_i = (y_{i+1} + y_{i-1})/2,$$

где yy_i – точки сглаженной кривой;

y_{i+1}, y_{i-1} – точки несглаженной кривой.

Таблица 5.14

Вариант	Функция
1	2
1	$\ln(x+5\pi) + \sin 4(x+\pi/4) + 80$
2	$\log_2(x+10\pi) + 5\cos(x-\pi/4) + \sin(x)$
3	$3\mathrm{tg}(x+\pi/10) + 4\cos\left(\frac{1}{x+1}\right) + \cos(x)$
4	$\sin[\cos^2(x+4\pi)] + \ln(x+8)$
5	$\cos(x) + 2\sin(x) + \cos(x)$
6	$\mathrm{tg}(x) + \mathrm{ctg}(x) + \mathrm{tg}(x)$
7	$4\cos(x) + x^2 + 2x$
8	$3\sin(x) + 0{,}5\sin^2(x) + 7$
9	$2 \cdot \log_4(x+12\pi) + 3 \cdot \log_5(x+13\pi)$
10	$\cos^3(x) + 2\cos^2(x) + 3\cos(x) + 7$
11	$\sin(3x) + \sin(6x) + \sin(9x) + \sin(11x)$
12	$\mathrm{tg}(x) + \mathrm{tg}(3x) + \mathrm{tg}(5x) + \mathrm{tg}(7x)$
13	$\mathrm{ctg}(x) + \mathrm{ctg}(2x) + \mathrm{ctg}(7x) + \mathrm{ctg}(9x)$
14	$\cos^3(x) + 2\cos^2(x) + 3\cos(x) + 7$
15	$\sin^4(x) + 4\sin^2(x) + 77$
16	$\mathrm{tg}^2(x) + \mathrm{tg}(x) + 2\mathrm{ctg}(x)$
17	$\mathrm{ctg}(x) + \mathrm{ctg}^3(x) + \mathrm{ctg}^2(x)$
18	$\sin^2(x) + \cos^3(x) + \sin(x+\pi)$
19	$\ln(x+5\pi) + \sin 4(x+\pi/4) + 80$

Продолжение табл. 5.14

1	2
20	$\log_2(x+10\pi)+5\cos(x-\pi/4)+\sin(x)$
21	$3\operatorname{tg}(x+\pi/10)+4\cos\left(\frac{1}{x+1}\right)+\cos(4x)$
22	$\sin[\cos^2(x+4\pi)]+\ln(x+8)$
23	$\cos(3x)+2\sin(6x)+\cos(4x)$
24	$\operatorname{tg}(4x)+\operatorname{ctg}(x)+\operatorname{tg}(11x)$
25	$4\cos(x)+x^2+2x$
26	$5\sin(11x)+2\sin^2(2x)+3$
27	$4\log_6(x+14\pi)+5\log_7(x+15\pi)$
28	$\cos^3(3x)+2\cos^2(3x)+3\cos(3x)+7$
29	$\sin(3x)+\sin(6x)+\sin(9x)+\sin(11x)$
30	$\operatorname{tg}(x)+\operatorname{tg}(3x)+\operatorname{tg}(5x)+\operatorname{tg}(7x)$
31	$\operatorname{ctg}(x)+\operatorname{ctg}(2x)+\operatorname{ctg}(7x)+\operatorname{ctg}(9x)$
32	$\cos^3(3x)+2\cos^2(3x)+3\cos(3x)+7$
33	$\sin^4(3x)+4\sin^2(3x)+77$
34	$\operatorname{tg}(x)+\operatorname{tg}^2(3x)+2\operatorname{ctg}(4x)$
35	$\operatorname{ctg}(4x)+\operatorname{ctg}^3(11x)+\operatorname{ctg}^2(3x)$
36	$\sin^2(4x)+\cos^3(x)+\sin(4x+\pi)$
37	$\ln(x+3\pi^3)+\sin 4(x-\pi/6)$
38	$\log_5(x+20\pi)+15\cos(x-\pi/4)+\sin(3x)$
39	$\frac{1}{3}\operatorname{tg}^2(x+\pi/10)+6\cos\left(\frac{1}{x+1}\right)+\sin(4x)$
40	$\sin[\cos^3(x+4\pi^2)]+\ln(x+18)$
41	$\cos(3x)+12\sin(6x)+5\cos(x)$

Окончание табл. 5.14

1	2
42	$4\mathrm{tg}(x)+7\mathrm{ctg}(1x)+11\mathrm{tg}(4x)$
43	$14\cos(6x)+x^3+5x$
44	$5\sin(1x)+2\sin^2(2x)+3$
45	$\log_4(+12\pi)+\log_5(+13\pi)+\log_6(+14\pi)$
46	$\cos^3(x)+2\cos^2(x)+3\cos(x)+7$
47	$2\sin(3x)+3\sin(6x)+4\sin(9x)+5\sin(11x)$
48	$2\mathrm{tg}(x)+3\mathrm{tg}(3x)+4\mathrm{tg}(5x)-5\mathrm{tg}(7x)$
49	$2\mathrm{ctg}(x)+3\mathrm{ctg}(2x)+4\mathrm{ctg}(7x)+5\mathrm{ctg}(9x)$
50	$\cos(x)+2\cos^2(x)+3\cos^3(x)+2$

5.14. **Задачи по разработке технологии интегрирования функций в** MS EXCEL

Постановка задачи. Вычислить значения суммы интегралов для различных значений параметра, используя метод трапеций (*а*), прямоугольников (*б*) и Симпсона (*в*). Варианты заданий представлены в табл. 5.15. Использовать пять шагов интегрирования, определяя их по формулам:

$$\Delta x_0 = \Delta x,$$

$$\Delta x_1 = \Delta x_0 / 4,$$

$$\Delta x_2 = \Delta x_1 / 4,$$

$$\Delta x_3 = \Delta x_2 / 4,$$

$$\Delta x_4 = \Delta x_3 / 4,$$

где Δx – начальное значения шага, соответствующее не менее 100 разбиениям интервала интегрирования.

Построить зависимость вычисляемого интеграла от шага интегрирования для различных параметров *k*.

Вид вычисляемой суммы интегралов

$$S = \int_{x_0}^{x_n} f(x \cdot k)\,dx + \int_{x_0}^{2x_n} f^2(x + k^2)\,dx.$$

Таблица 5.15

Номер варианта	Подынтегральная функция $f(x)$	x_0	x_n	Метод	Параметр k
1	2	3	4	5	6
1	$e^x \cdot \cos(4x)$	2	10	а	1, 2, 3
2	$e^{x^4} \cdot \sin\left(x + \pi/4\right)$	4	7	б	4, 5, 6
3	$x^7 \cdot \sin(x + \pi)$	3	20	в	10, 11, 12
4	$\cos(8x) \cdot \mathrm{tg}(9x)$	7	23	а	7, 9, 12
5	$\mathrm{arctg}(9x) + \sin(4x)$	8	15	б	8, 10, 12
6	$\ln(8x) + \log_2(9x)$	9	22	в	9, 13, 17
7	$x^2 \cdot \ln(9x)$	12	17	а	13, 14, 15
8	$e^{3x} \cdot \cos(4x)$	1	10	б	18, 20, 22
9	$4\sin(7x^2) \cdot \mathrm{tg}(18x)$	11	21	в	24, 26, 28
10	$5\cos(10x) \cdot \ln(7x)$	24	40	а	30, 32, 34
11	$\ln(8x) \cdot (x^2 + 5x + 3)$	30	70	б	37, 40, 43
12	$(x^2 + 10x) \cdot \cos\left(x + \pi/10\right)$	12	20	в	8, 10, 12
13	$e^{3x} \cdot x^2 \cdot \cos(x^2)$	14	30	а	13, 15, 17
14	$x^2 \cdot \ln(3x) \cdot \sin(3x)$	15	31	б	20, 22, 24
15	$x \cdot \ln(x) \cdot (x^2 + 2x + x^3)$	17	33	в	70, 73, 76
16	$(x^2 + 10x + \cos(4x)) \cdot x$	18	28	а	80, 84, 88
17	$x^8 + \cos^2(4x) + e^{4x}$	30	40	б	92, 95, 97
18	$x \cdot \cos^4(4x) + x^{2,5}$	11	21	в	100, 102, 104

Продолжение табл. 5.15

1	2	3	4	5	6
19	$x^{2/3}\cdot\cos^2(4x)$	24	34	а	33, 36, 39
20	$e^x\cdot\cos^3(7x)+x^4$	40	50	б	31, 32, 33
21	$x^2\cdot\cos(4x)+x^4\cdot\sin(7x)$	60	80	в	41, 42, 43
22	$x\cdot\cos(7x)+x^3\cdot\sin(8x)$	70	90	а	70, 71, 72
23	$x\cdot\mathrm{tg}(x^2)+x^3\cdot\mathrm{ctg}(x^3)$	100	110	б	81, 84, 87
24	$e^{2x}\cdot\cos(4x^2)$	2	10	а	13, 15, 17
25	$e^{x^3}\cdot\sin\left(x^{1/2}+\pi/2\right)$	4	7	б	20, 22, 24
26	$x^5\cdot\sin\left(6x+\pi\right)$	3	20	в	70, 73, 76
27	$\cos(x^3)\cdot\mathrm{tg}(9x^{1/3})$	7	23	а	80, 84, 88
28	$\mathrm{tg}(9x)+\sin(4x^{1/4})$	8	15	б	92, 95, 97
29	$\ln(5x^{1/5})+\log_6(9x)$	9	22	в	100, 102, 104
30	$x^{1/2}\cdot\ln(4x^4)$	12	17	а	33, 36, 39
31	$e^{4x}\cdot\cos(x^{1/5})$	1	10	б	31, 32, 33
32	$8\sin(8x^{1/8})\cdot\mathrm{tg}(8x^{1/8})$	11	21	в	41, 42, 43
33	$\cos(10x^{0,1})\cdot\ln(7x^{1/7})$	24	40	а	70, 71, 72
34	$\ln x^{1/8}\cdot\left(x^2+10x+0,3\right)$	30	70	б	81, 84, 87
35	$\left(x^{1/2}+18x\right)\cdot\cos\left(x+\pi/80\right)$	12	20	в	13, 14, 15
36	$e^{2x}\cdot x^{1/2}\cdot\cos(x^3)$	14	30	а	18, 20, 22
37	$x^3\cdot\ln(x^2)\cdot\sin(5x)$	15	31	б	24, 26, 28
38	$x^3\cdot\ln(x^2)\cdot\left(x^2+4x+x^{1/3}\right)$	17	33	в	30, 32, 34
39	$\left(x^2+5x+\cos x^4\right)\cdot x^{1/4}$	18	28	а	37, 40, 43
40	$x^7+\cos^{1/2}(8x)+e^{3x}$	30	40	б	8, 10, 12

Окончание табл. 5.15

1	2	3	4	5	6
41	$x^{1/3}\cdot\cos^5(4x)+x^{3,5}$	11	21	в	13, 15, 17
42	$x^{2/3}\cdot\cos^2(4x)$	24	34	а	20, 22, 24
43	$e^{1/x}\cdot\cos^5(5x)+x^3$	40	50	б	1, 2, 3
44	$x^{1/2}\cdot\cos(4x^4)+x^{1/4}\cdot\sin(17x)$	60	80	в	4, 5, 6
45	$x^2\cdot\cos(17x)+x^{1/3}\cdot\sin(18x)$	70	90	а	10, 11, 12
46	$x^3\cdot\mathrm{tg}(x^2)+x^{1/3}\cdot\mathrm{ctg}(x^2)$	100	110	б	7, 9, 12
47	$e^{1/x}\cdot\cos(4x^2)$	2	10	а	24, 26, 28
48	$e^{x^3}\cdot\sin\left(x^{1/4}+\pi/4\right)$	4	7	б	30, 32, 34
49	$x^{1/7}\cdot\sin(7x+\pi)$	3	20	в	37, 40, 43
50	$\cos(18x)\cdot\mathrm{tg}(x^2)$	7	23	а	8, 10, 12

Пример для первого варианта. Используя метод трапеций, вычислить интегралы.

$$S=\int_2^{10} f(x\cdot 1)\,dx+\int_2^{20} f^2(x+1^2)\,dx,$$

$$SS=\int_2^{10} f(x\cdot 2)\,dx+\int_2^{20} f^2(x+2^2)\,dx,$$

$$SSS=\int_2^{10} f(x\cdot 3)\,dx+\int_2^{20} f^2(x+3^2)\,dx,$$

где вид подынтегральной функции $f(x)=e^x\cdot\cos 4x$.

Для каждого интеграла вычислять значения для пяти различных шагов. При выполнении задания заполнить таблицу (табл. 5.16) и исследовать влияние шага разбиения на величину интеграла.

Таблица 5.16

$S(\Delta x_0 = \Delta x)$	$SS(\Delta x_0 = \Delta x)$	$SSS(\Delta x_0 = \Delta x)$
$S(\Delta x_1 = \Delta x_0/4)$	$SS(\Delta x_1 = \Delta x_0/4)$	$SSS(\Delta x_1 = \Delta x_0/4)$
$S(\Delta x_2 = \Delta x_1/4)$	$SS(\Delta x_2 = \Delta x_1/4)$	$SSS(\Delta x_2 = \Delta x_1/4)$
$S(\Delta x_3 = \Delta x_2/4)$	$SS(\Delta x_3 = \Delta x_2/4)$	$SSS(\Delta x_3 = \Delta x_2/4)$
$S(\Delta x_4 = \Delta x_3/4)$	$SS(\Delta x_4 = \Delta x_3/4)$	$SSS(\Delta x_4 = \Delta x_3/4)$

5.15. **Задачи по разработке технологии решения дифференциальных уравнений методом Рунге-Кутта в** MS EXCEL

Постановка задачи. Разработать алгоритм решения методом Рунге-Кутта дифференциального уравнения $y' = f(x, y)$ на интервале отрезка $[a, b]$ при начальном условии $y(x = a) = c$. Шаг изменения переменной выбрать самостоятельно. Варианты заданий представлены в табл. 5.17.

Таблица 5.17

Номер варианта	Дифференциальное уравнение	a	b	c
1	2	3	4	5
1	$e^{y} \cdot \cos(4x)$	2	10	22
2	$e^{x^4} \cdot \sin(4y + \pi/4)$	4	7	4
3	$y \cdot \sin(x + \pi)$	3	20	10
4	$\cos(8y) \cdot \mathrm{tg}(9x)$	7	23	7
5	$\mathrm{arctg}(9y) + \sin(4x)$	8	15	8
6	$\ln(8y) + \log_2(9x)$	9	22	9
7	$x^2 \cdot \ln(9x)$	12	17	13
8	$e^{3y} \cdot \cos(4x)$	1	10	18
9	$\sin(7y^2) \cdot \mathrm{tg}(8x)$	11	21	24
10	$\cos(y) \cdot \ln(7x)$	24	40	30

Продолжение табл. 5.17

1	2	3	4	5
11	$\ln(y)\cdot(x^2+5x+3)$	30	70	37
12	$(y^2+10x)\cdot\cos(4x+\pi/10)$	12	10	22
13	$e^{3y}\cdot x^2\cdot\cos(x^2)$	4	7	4
14	$y^2\cdot\ln(3x)\cdot\sin(3x)$	5	20	10
15	$y\cdot\ln(x)\cdot(x^2+2x+x^3)$	7	23	7
16	$(y^2+10x+\cos(4x))\cdot x$	1	15	8
17	$y+\cos^2(4x)+e^{4x}$	3	22	9
18	$y\cdot\cos^4(4x)+x^{2,5}$	11	17	13
19	$y^{2/3}\cdot\cos^2(4x)$	2	10	18
20	$e^y\cdot\cos^3(7x)+x^4$	4	21	24
21	$y^2\cdot\cos(4x)+x\cdot\sin(7x)$	6	40	30
22	$y\cdot\cos(7y)+x^3\cdot\sin(8x)$	7	10	22
23	$y\cdot\mathrm{tg}(y^2)+x\cdot\mathrm{ctg}(x^3)$	1	7	4
24	$e^{2y}\cdot\cos(4x^2)$	2	20	10
25	$e^{y^3}\cdot\sin(4x^{1/2}+\pi/2)$	4	23	7
26	$y\cdot\sin(6x+\pi)$	3	15	8
27	$\cos(y^3)\cdot\mathrm{tg}(9x^{1/3})$	7	22	9
28	$\mathrm{tg}(9y)+\sin(4x^{1/4})$	8	17	13
29	$\ln(5y^{1/5})+\log_6(9x)$	9	10	18
30	$y^{1/2}\cdot\ln(4x^4)$	1	10	22
31	$e^{4y}+\cos(x^{1/5})$	1	7	4
32	$8\sin(8y)\cdot\mathrm{tg}(8x^{1/8})$	1	20	10
33	$\cos(10y)\cdot\ln(7x^{1/7})$	2	23	7

Окончание табл. 5.17

1	2	3	4	5
34	$\ln(y^{1/8})\cdot(x^2+10x+0{,}3)$	3	15	8
35	$(x^{1/2}+18y)\cdot\cos(4y+\pi/80)$	1	22	9
36	$e^{2y}\cdot\cos(x^3)$	4	17	13
37	$y^3\cdot\sin(5x)$	5	10	18
38	$x^3\cdot(y^2+4x)$	7	21	24
39	$(y^2+5y+\cos(x^4))\cdot x^{1/4}$	8	40	30
40	$y+\cos^{1/2}(8x)+e^{3x}$	30	70	37
41	$\cos^5(4y)+x^{3,5}$	11	21	17
42	$x^{2/3}\cdot\cos^2(4y)$	24	34	24
43	$\cos^5(5yx)+x^3$	40	50	3
44	$x^{1/2}\cdot\cos(4y^4)$	60	80	6
45	$x^{1/3}\cdot\sin(18yx)$	70	90	12
46	$x^3\cdot\mathrm{tg}(y^2)$	1	11	12
47	$e^{1/y}\cdot\cos(4x^2)$	2	10	28
48	$e^y\cdot\sin(4x^{1/4}+\pi/4)$	4	7	34
49	$y\cdot\sin(7x+\pi)$	3	20	43
50	$\cos(18y)\cdot\mathrm{tg}(x^2)$	7	23	12

Построить решение дифференциального уравнения $y = F(x)$.

5.16. **Задачи по разработке технологии решения задачи оптимизации методом простых итераций в** MS EXCEL

Постановка задачи. Определить комбинацию значений переменных $x_1, x_2, \ldots, x_n$, для которых выполняются условия

$$\begin{cases} a_{11}\cdot f_1(x_1)+a_{12}\cdot f_2(x_2)+\ldots+a_{1j}\cdot f_j(x_j)+\ldots+a_{1n}\cdot f_n(x_n)=c_1\to F_1, \\ a_{21}\cdot f_1(x_1)+a_{22}\cdot f_2(x_2)+\ldots+a_{2j}\cdot f_j(x_j)+\ldots+a_{2n}\cdot f_n(x_n)=c_2\to F_2, \\ \ldots \\ a_{m1}\cdot f_1(x_1)+a_{m2}\cdot f_2(x_2)+\ldots+a_{mj}\cdot f_j(x_j)+\ldots+a_{mn}\cdot f_n(x_n)=c_m\to F_m, \end{cases}$$

где $F_1(x_1, x_2, \ldots, x_n) = 0$;

$F_2(x_1, x_2, \ldots, x_n) = 0$;

…

$F_m(x_1, x_2, \ldots, x_n) = 0$;

n – число переменных;

m – число уравнений.

Функционал цели выбрать на основе условия $F = (F_1, F_2, \ldots, F_m) \to \min$. Значения n, m и вид функционала цели заданы вариантом задачи.

Коэффициенты системы уравнений вычисляются по законам

$$a_{ij} = n\cdot m - i - j,\ \ c_i = \ln(i\cdot m\cdot k).$$

Предположить следующие интервалы изменения переменных: x_1 = [5; 15], x_2 = [0; 40], x_3 = [–10; 20], x_4 = [0; 20], x_5 = [10; 25], x_6 = [–0,5; 3,4]. При формировании функционала цели учесть область определения.

Значения n, m и вид функционала цели F выбрать из табл. 5.18, функции $f_i(x_i)$ – из табл. 5.19 в соответствии с вариантом задачи.

Таблица 5.18

Номер варианта	n	m	Функционал цели $F = (F_1, F_2, \ldots, F_m)$
1	2	3	4
1	3	3	$F_1^2 + F_2^2 + F_3^2$
2	3	4	$F_1^2 + F_2^2 + \sqrt{F_3} - F_4$
3	3	5	$F_1^3 + \dfrac{1}{F_2^2 + 1} - 2F_3 + F_4 + F_5$
4	3	6	$3F_1 + 2F_2^2 - 2F_3 + F_4^2 + F_5 - 0{,}5F_6$

Продолжение табл. 5.18

1	2	3	4
5	4	2	$3F_1^3 - 5F_2^2$
6	4	3	$3F_1 + 2F_2^2 - 0{,}5F_3$
7	4	4	$\frac{1}{3}F_1^2 + F_2^2 - \frac{F_3}{F_4}$
8	4	5	$F_1 + F_2^2 - F_3^2 + \frac{1}{F_4 + F_5}$
9	4	6	$F_1 \cdot F_2^2 - F_3^2 \cdot F_4 + F_5 \cdot F_6$
10	5	2	$\frac{1}{F_1^2 + 7} - F_2^2$
11	5	3	$3F_1 \cdot F_2^2 - 0{,}5F_3$
12	5	4	$F_1^2 - F_2^2 + \sqrt{F_3} - F_4$
13	5	5	$\frac{1}{7}F_1^2 + F_2^2 - \frac{F_3}{F_4} - F_5^2$
14	5	6	$F_1^2 - F_2^2 \cdot \sqrt{F_3} - F_4 \cdot F_5 + F_6$
15	6	2	$F_1^2 + F_2^2$
16	6	3	$F_1^2 + F_2^2 + F_3$
17	6	4	$\sqrt{F_1} + F_2^2 - 3F_3 + F_4$
18	6	5	$F_1^2 + F_2^2 + \sqrt{F_3} - 2F_4 + 0{,}3F_5$
19	6	6	$F_1 + F_2^2 - \frac{F_3^2}{F_4} + F_5 \cdot F_6$
20	3	3	$F_1^2 - F_2^2 + 4F_3$
21	3	4	$\sqrt{F_1} \cdot F_2^2 + 0{,}5F_3 - 2F_4$
22	3	5	$F_1 - 2F_2^2 + F_3^2 + \frac{1}{F_4 \cdot F_5}$
23	3	6	$F_1^2 + F_2^2 + \sqrt{F_3} + F_4 + F_5 + F_6$

Продолжение табл. 5.18

1	2	3	4
24	4	2	$-\frac{1}{9}F_1^2+F_2^3$
25	4	3	$F_1^2-5F_2^2+2F_3$
26	4	4	$\sqrt{F_1}+F_2^2-3F_3+F_4$
27	4	5	$5\cdot\sqrt{F_1}+F_2^2-\frac{F_4}{3F_3}+F_5$
28	4	6	$\sqrt{F_1}\cdot F_6+F_2^2-3F_3+F_4\cdot F_5$
29	5	2	$7F_1-2F_2$
30	5	3	$5F_1-2F_2^2+F_3^2$
31	5	4	$F_1-2F_2^2+F_3^2+\sqrt{\frac{1}{F_4}}$
32	5	5	$F_1^2-2F_2^2+F_3-F_4-F_5$
33	5	6	$7\cdot\sqrt{F_1}-2\cdot F_2^2-\frac{F_4\cdot F_5}{5F_3}$
34	6	2	$7\cdot\sqrt{F_1}-9\cdot F_2^2$
35	6	3	$\sqrt{F_1}-2\cdot F_2^2-\frac{1}{3F_3}$
36	6	4	$F_1^2\cdot F_4-5F_2^2+2F_3$
37	6	5	$F_1-2F_2+F_3^2+F_4\cdot F_5$
38	6	6	$F_1+2F_2+F_3+F_4+F_5\cdot F_6$
39	3	3	$F_1+2F_2-F_3$
40	3	4	$F_1-2F_2\cdot F_3+\sqrt{F_4}$
41	3	5	$-\frac{F_1}{2F_2+F_3}+F_4-3F_5$
42	3	6	$F_1^2+F_2^2+\sqrt{F_3}\cdot F_4+F_5\cdot F_6$
43	4	2	$F_1^2-F_2^2$

Окончание табл. 5.18

1	2	3	4
44	4	3	$F_1^2 - F_2^2 + F_3^2$
45	4	4	$F_1^2 - F_2^2 + F_3^2 + F_4$
46	4	5	$F_1^2 - F_2^2 + F_3^2 - F_4^2 + F_5$
47	4	6	$F_1^2 - F_2^2 + F_3^2 - F_4^2 - F_5^2 + F_6$
48	5	2	$F_1^2 + 5F_2$
49	5	3	$-F_1^2 + F_2^2 + F_3^2$
50	5	4	$F_1 - 2F_2 + \frac{\sqrt{F_4}}{F_3}$

Таблица 5.19

i	1	2	3	4	5	6
$f_i(x_i)$	$\cos(4 \cdot x_1)$	$\operatorname{tg}(8 \cdot x_2)$	$\ln(4 \cdot x_3)$	$e^{-x_4^2}$	$\frac{1}{x_5^2 + 7}$	$\log_2(4 \cdot x_6)$

Например, для варианта 2 условие задачи с учетом данных в табл. 5.18, 5.19 будет иметь вид:

Шаг 1. Вычисление коэффициентов системы уравнений:

$a_{11} = 3 \cdot 4 - 1 - 1 = 10,\ a_{12} = 3 \cdot 4 - 1 - 2 = 9,\ a_{13} = 3 \cdot 4 - 1 - 3 = 8;\ c_1 = \ln(1 \cdot 3 \cdot 4) = \ln(12);$
$a_{21} = 3 \cdot 4 - 2 - 1 = 9,\ a_{22} = 3 \cdot 4 - 2 - 2 = 8,\ a_{23} = 3 \cdot 4 - 2 - 3 = 7;\ c_2 = \ln(2 \cdot 3 \cdot 4) = \ln(24);$
$a_{31} = 3 \cdot 4 - 3 - 1 = 8,\ a_{32} = 3 \cdot 4 - 3 - 2 = 7,\ a_{33} = 3 \cdot 4 - 3 - 3 = 6;\ c_3 = \ln(3 \cdot 3 \cdot 4) = \ln(36);$
$a_{41} = 3 \cdot 4 - 4 - 1 = 7,\ a_{42} = 3 \cdot 4 - 4 - 2 = 6,\ a_{43} = 3 \cdot 4 - 4 - 3 = 5;\ c_4 = \ln(4 \cdot 3 \cdot 4) = \ln(48).$

Шаг 2. Формирование системы уравнений

$$\begin{cases} a_{11} \cdot f_1(x_1) + a_{12} \cdot f_2(x_2) + a_{13} \cdot f_3(x_3) = c_1 \to F_1, \\ a_{21} \cdot f_1(x_1) + a_{22} \cdot f_2(x_2) + a_{23} \cdot f_3(x_3) = c_2 \to F_2, \\ a_{31} \cdot f_1(x_1) + a_{32} \cdot f_2(x_2) + a_{33} \cdot f_3(x_3) = c_3 \to F_3, \\ a_{41} \cdot f_1(x_1) + a_{42} \cdot f_2(x_2) + a_{43} \cdot f_3(x_3) = c_4 \to F_4. \end{cases}$$

После подстановки значений коэффициентов a_{ij}, c_i и функций $f_i(x_i)$ система будет иметь вид

$$\begin{cases} 10\cdot\cos 4\cdot x_1 + 9\cdot \mathrm{tg}\, 8x_2 + 8\cdot \ln 4x_3 = \ln(12) \to F_1, \\ 9\cdot\cos 4\cdot x_1 + 8\cdot \mathrm{tg}\, 8x_2 + 7\cdot \ln 4x_3 = \ln(24) \to F_2, \\ 8\cdot\cos 4\cdot x_1 + 7\cdot \mathrm{tg}\, 8x_2 + 6\cdot \ln 4x_3 = \ln(36) \to F_3, \\ 7\cdot\cos 4\cdot x_1 + 6\cdot \mathrm{tg}\, 8x_2 + 5\cdot \ln 4x_3 = \ln(48) \to F_4. \end{cases}$$

Шаг 3. Запись функционала цели

$$F = F_1^2 + F_2^2 + \sqrt{F_3} - F_4 \to \min.$$

Шаг 4. Формирование ограничений

$$x_1 = [5; 15],\ x_2 = [0; 40],\ x_3 = [-10; 20],\ F_3 \geq 0.$$

РЕКОМЕНДУЕМЫЕ УЧЕБНЫЕ ПОСОБИЯ ДЛЯ СТУДЕНТОВ

1. Макконелл, Джефри. Анализ алгоритмов: вводный курс / Джефри Макконелл; пер. С.К. Ландо. – М.: Техносфера, 2002. – 302 с.

2. Макконелл, Джефри. Основы современных алгоритмов / Джефри Макконелл; ред. перевода С.К. Ландо. – 2-е изд., доп. – М.: Техносфера, 2006. – 366 с.

3. Федоренко, Ю.В. Алгоритмы и программы на Turbo Pascal / Ю.В. Федоренко. – СПб.: Питер, 2001. – 240 с.

4. Жернак, А.Н. Программирование. Основы конструкции и операторы языка Паскаль: текст лекций / А.Н. Жернак, Р.А. Кимашева, М.В. Копейкин. – Л.: СЗПИ, 1991. – 48 с.

5. Фаронов, В.В. Турбо Паскаль [7.0]. Начальный курс: учебное пособие / В.В. Фаронов. – М.: Нолидж, 1999. – 612 с.

6. Фаронов, В.В. Турбо Паскаль 7.0: Практика программирования: учебное пособие / В.В. Фаронов. – М.: Нолидж, 1999. – 429 с.

7. Могилев, А.В. Информатика / А.В. Могилев, Н.И. Пак, Е.К. Хеннер. – М.: Издательский центр «Академия», 2004. – 848 с.

8. Ставровский, А.Б. Турбо Паскаль 7.0: учебник / А.Б. Ставровский. – М.: Нолидж, 2000. – 399 с.

9. Вальвачев, А.Н. Программирование на языке Паскаль для персональных ЭВМ ЕС: справочное пособие / А.Н. Вальвачев, В.С. Крисевич. – М.: Высшая школа, 1989. – 223 с.

10. Мудров, А.Э. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль / А.Э. Мудров. – Томск: МП «Раско», 1991. – 272 с.

11. Калиткин, Н.Н. Численные методы / Н.Н. Калиткин. – М.: Наука, 1978. – 512 с.

12. Волков, Е.А. Численные методы / Е.А. Волков. – М.: Наука, 1987. – 248 с.

13. Чичко, О.И. Алгоритмы и технология решения задач матричного исчисления в MS EXCEL: учебное пособие / О.И. Чичко, В.И. Махнач, А.Н. Чичко. – Минск: БНТУ, 2005. – 105 с.

14. Чичко, А.Н. Информатика / А.Н. Чичко, Е.А. Дроздов. – Минск: БНТУ, 2000. – 151 с.

15. Чичко, О.И. Алгоритмы и технология решения численных задач в MS EXCEL / О.И. Чичко, В.И. Махнач, А.Н. Чичко. – Минск: БНТУ, 2005. – 101 с.

16. Чичко, А.Н. Информатика: учебно-методическое пособие / А.Н. Чичко, А.С. Бороздин. – Минск: БНТУ, 2003. – 187 с.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1
КРАТКИЕ СВЕДЕНИЯ О ЯЗЫКЕ PASCAL

П 1.1. Алфавит языка Pascal

Язык Pascal включает следующий набор основных символов:
1) 26 латинских строчных и прописных букв:
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z;
2) символ подчеркивание «_»;
3) 10 цифр (0 1 2 3 4 5 6 7 8 9);
4) знаки операций (+ - * / = <> < > <= >= := @);
5) ограничители: (. , ' ( ) [ ] (. .) {} (* *) .. :);
6) спецификаторы: (^ # $);
7) служебные (зарезервированные) слова:

ABSOLUTE	FILE	MOD	SHR
AND	FOR	NIL	STRING
ARRAY	FORWARD	NOT	THEN
BEGIN	FUNCTION	OF	TO
CASE	GOTO	OR	TYPE
CONST	IF	PACKED	UNIT
DIV	IMPLEMENTATION	PROCEDURE	UNTIL
DO	IN	PROGRAM	USES
DOWNTO	INLINE	RECORD	VAR
ELSE	INTERFACE	REPEAT	WHILE
END	INTERRUPT	SET	WITH
EXTERNAL	LABEL	SHL	XOR

Кроме того, в набор основных символов входит пробел « ».

П 1.2. Элементарные конструкции

Элементарные конструкции языка Pascal включают в себя имена, числа и строки.

Имена (идентификаторы) – это элементы языка: константы, метки, переменные, процедуры, функции и т. д. *Имя* – это последовательность букв и цифр, начинающаяся с буквы. В именах может использоваться символ «подчеркивание». В качестве имен не разрешается использовать служебные слова и стандартные имена, которыми названы стандартные константы, типы, процедуры, функции и файлы.

Примеры правильных имен языка Pascal*:*

M, bFlag, iMax, OutFile, F_1

Примеры неправильных имен языка Pascal*:*

1qwick, Begin, function

Числа в языке Pascal обычно записываются в десятичной системе счисления. Они могут быть целыми и действительными. Действительные числа записываются в форме с десятичной точкой или в форме с использованием десятичного порядка, который изображается буквой E:

32.1, -0.06, 3E7, 2.3E-12.

Строки в языке Pascal – это последовательность символов, записанных между апострофами:

‘Строка 1’, ‘Введите a’, ‘Значение функции = ’.

Выражение состоит из констант, переменных, указателей функций, знаков операций и скобок. Выражение задает правило вычисления некоторого значения. Порядок вычисления определяется старшинством (приоритетом) содержащихся в нем операций.

Основные операции языка Pascal представлены в табл. П 1.1. Для изменения приоритета операций необходимо использовать скобки.

Таблица П 1.1

Операция	Описание	Пример	Приоритет
=	Равно	A = B	4
+	Сложить	A + B	3
–	Вычесть	A – B	3
/	Делить	A / B	2
*	Умножить	A * B	2
<>	Не равно	A <> B	4
<	Меньше	A < B	4
>	Больше	A > B	4
not	Отрицание	not A	1
and	Логическое «и»	A and B	2

or	Логическое «или»	A or B	3

В табл. П 1.2 приведены некоторые математические функции языка Pascal, использованные в заданиях и примерах учебного пособия. Наиболее высокий приоритет имеет операция отрицания, затем следующий уровень приоритета имеют операции «умножить», «делить», логическое «и». Самый малый уровень приоритета имеет операция «равно».

Таблица П 1.2

Функция	Назначение
exp(*x*)	Вычисление экспоненты *x*
ln(*x*)	Вычисление натурального логарифма *x*
abs(*x*)	Вычисление абсолютной величины *x*
sqr(*x*)	Вычисление квадрата величины *x*
sqrt(*x*)	Извлечение корня *x*
trunc(*x*)	Получение целой части значения *x* (результат Integer)
round(*x*)	Округление в сторону ближайшего целого *x*
odd(*x*)	True – если *x* – нечетное и False – если *x* – четное
frac(*x*)	Вычисление дробной части *x*
int(*x*)	Вычисление целой части *x* (результат Real)
pi	Возвращает число π (пи)
sin(*x*)	Вычисление синуса *x*
cos(*x*)	Вычисление косинуса *x*
arctan(*x*)	Вычисление арктангенса *x*

В языке Pascal имеются минимальные возможности для вычисления функций, поэтому сложные функции строятся на основе элементарных. При использовании сложных тригонометрических и логарифмических функций рекомендуется использовать следующие формулы:

$\arcsin(x) = \operatorname{arctg}\left[(1-x^2)^{1/2}\right]$, если $x(1-x^2) \geq 0$;

arccos(*x*) = *pi*/2 – arcsin(*x*);

arctctg(*x*) = *pi*/2 – arctg(*x*);

sec(*x*) = 1/cos(*x*), если cos(*x*) ≠ 0;

tg(*x*) = sin(*x*)/cos(*x*), если cos(*x*) ≠ 0;

$\log_a(x) = \ln(x)/\ln(a)$;

$x^A = e^{\ln x^A} = \exp(A * \ln(x))$, если $x > 0$.

Пример. Записать фрагмент выражения $y = (\cos(x)+1)^7$ в языке Pascal.

```
…
Y := cos(x) + 1;
Y :=exp(7*ln(Y));
…
```

или

```
Y := exp(7*ln((cos(x)+1))
```

Пример. Записать выражение $y = \ln^{7/3}(\cos^3(x))$ в языке Pascal с учетом области определения функции.

```
If (cos(x) > 0) and (Ln(cos(x)* cos(x)* cos(x)) > 0) then
                  Y := exp(7/3*ln(ln(cos(x)*cos(x)*cos(x))));
```

П 1.3. Структура программы на языке Pascal

Структура программы на языке Pascal имеет вид

```
Program     <имя программы>;
Uses   <подключаемые модули>;
Label {раздел описания меток, используемых в программе }
Const {раздел описания констант, используемых в программе}
Type  {раздел описания типов, используемых в программе}
Var  {раздел описания переменных, используемых в программе}
{раздел описания процедур и функций}
Procedure  <имя процедуры>;
      <тело процедуры>;
Function     <имя функции>;
      <тело функции>;
 {раздел операторов основной программы}
Begin
<операторы>;
End.
```

После зарезервированного слова Program идет название программы. Любой раздел, кроме раздела операторов, может отсутствовать. Все используемые объекты программы должны быть описаны в соответствующих разделах.

Рассмотрим примеры использования различных элементов программы:

```
Uses Crt, Dos;{подключение библиотечных модулей crt и dos}
Const {раздел описания констант}
   Point_beg =3.80, Point_End = 130, M = 3.1415, M1 = 100;
Type {раздел описания типов}
   {описание типа Mas, как двухмерного массива}
   Mas = Array [1..20,1..40] of Real;
   {описание типа MM как одномерного массива размером 40}
   MM = Array[ 1..40] of Integer;
Var  {раздел описания переменных}
   A,B,C : Real; {описание переменных вещественного типа}
   i : Integer;  {описание переменной i - целого типа}
   k : Byte;     {описание переменной k - байтового типа}
```

П 1.4. Типы данных

Каждая константа, переменная, выражение или функция в языке Pascal имеет свой *тип*. *Тип* определяет множество допустимых значений, а также операции и функции, которые могут быть применимы к переменным, константам и другим величинам, принадлежащим данному типу. Типы в языке Pascal могут быть простыми и структурированными. Простые типы подразделяются на стандартные и пользовательские. Стандартными типами являются целые, вещественные, логический тип (Boolean), символьный (Char). К пользовательским типам относятся перечисляемый и интервальный типы. Представителями структурированных типов в языке Pascal являются массивы, файлы, записи и множества.

Описание переменной производится перед ее использованием в разделе описания переменных и имеет следующую форму записи:

```
Var   <имя переменной> : <тип>;.
```

Целые типы

Характеристики целых типов данных представлены в табл. П 1.3.

Таблица П 1.3

Тип	Требуемая память (байт)	Диапазон значений
Shortint	1	–128–127
Integer	2	–32 768–32 767
Longint	4	–2 147 483 648–2 147 483 647
Byte	1	0–255
Word	2	0–65 535

Пример описания переменных

```
Var
   I, J    : Byte;     {целые переменные типа Byte}
   Temp, T : Integer;  {целые переменные типа Integer}
   MaxA    : Word;     {целая переменная типа Word}
```

Вещественные типы

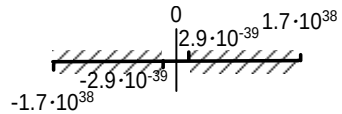
Числа действительного типа могут быть представлены в двух видах: числом с фиксированной и с плавающей точкой (экспоненциальная форма). Число с фиксированной точкой изображается десятичным числом с дробной частью (дробная часть может быть нулевой), например 127.3, 25.0, -16.003, 200.59, 0.54. Число с плавающей точкой имеет вид <*мантисса*>*E*<*порядок*>, где <*мантисса*> – вещественное число в форме с фиксированной точкой, <*порядок*> – целое число. Буква *E* может быть строчной или заглавной. Как мантисса, так и порядок могут содержать знаки «+» и «–». Эту запись следует понимать как произведение мантиссы на 10 в степени, равной порядку. Примеры записи действительных чисел приведены в табл. П 1.4.

Таблица П 1.4

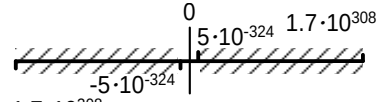
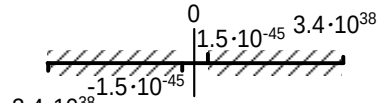
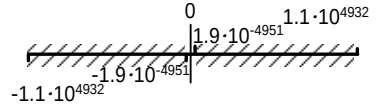
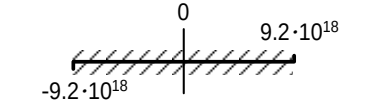
Математическая запись	Запись с плавающей точкой
0,000009	9E–6
$0{,}62\cdot10^{4}$	0.62E+4
$-10{,}8\cdot10^{12}$	–10.8E12
$20\cdot10^{-3}$	20E-3

В табл. П 1.5 представлены характеристики вещественных типов данных.

Таблица П 1.5

Тип	Память (байт)	Число значащих цифр	Область допустимых значений
Real	6	11–12	 $[-1{,}7\cdot10^{38};-2{,}9\cdot10^{-39}]\cup$ $\cup[2{,}9\cdot10^{-39};1{,}7\cdot10^{38}]$

Окончание табл. П 1.5

Тип	Память (байт)	Число значащих цифр	Область допустимых значений
Double	8	15–16	 $[-1{,}7\cdot10^{308};-5\cdot10^{-324}]\cup$ $\cup[5\cdot10^{-324};1{,}7\cdot10^{308}]$
Single	4	7–8	 $[-3{,}4\cdot10^{38};-1{,}5\cdot10^{-45}]\cup$ $\cup[1{,}5\cdot10^{-45};3{,}4\cdot10^{38}]$
Extended	10	19–20	 $[-1{,}1\cdot10^{4932};-1{,}9\cdot10^{-4951}]\cup$ $\cup[1{,}9\cdot10^{-4951};1{,}1\cdot10^{1932}]$
Comp	8		 $-2^{63}+1\ ..\ 2^{63}-1$ $[-9{,}2\cdot10^{18};9{,}2\cdot10^{18}]$

Примечание:

1) во всех указанных дипазонах нуль входит в область значений типа;

2) тип Comp содержит только целочисленные значения в диапазоне от $-2^{63}+1$ до $2^{63}-1$, что приблизительно равно $-9{,}2\cdot10^{18}$ и $9{,}2\cdot10^{18}$.

Например, числа типа Real могут принимать как положительные, нулевые, так и отрицательные значения и изменяться в

диапазоне от $-1{,}7\cdot10^{38}$ до $1{,}7\cdot10^{38}$, минимальным ненулевым абсолютным значением для них является $2{,}9\cdot10^{-39}=\frac{2{,}9}{10^{39}}$. Они имеют точность 11–12 знаков, т. е. числа, отличающиеся в 13 знаков, будут представлены одинаково.

Пример описания переменных

```
Var
  A, b : Real;     {вещественные переменные типа Real}
  P_1 : Double;    {вещественная переменная типа Double}
  e   : Extended;  {вещественная переменная типа Extended}
```

Логический тип (Boolean)

Логические типы переменных имеют два значения: истина (True) и ложь (False), занимают один байт памяти.

Пример описания переменных

```
Var  b1, D : Boolean;
```

Символьный тип (Char)

Символьные типы переменных Char могут принимать значения из множества символов ASCII, занимают один байт памяти.

Пример описания переменных

```
Var  C1, c2 : Char;
```

Строковый тип (String)

Для описания строки символов используется тип `String`:

```
<имя переменной> : String [<длина строки>];
```

где параметр `<длина строки>` указывает значение максимально допустимой длины строки. Параметр `<длина строки>` может быть опущен, тогда максимальная длина строки равна 255 символам.

Пример описания переменных

```
Var
   S1   : String [100]; {строка, состоящая из 100 символов}
   Str2 : String [12];  {строка, состоящая из 12 символов}
```

Массивы

Массивы – это формальное объединение нескольких однотипных объектов, рассматриваемых как единое целое.

Описание одномерного массива:

```
   Var
<имя массива> : Array [Нач_индекс..Кон_индекс] of <тип элементов>;
```

где параметры `Нач_индекс`, `Кон_индекс` указывают начальное и конечное значение индексов массива.

Доступ к элементу одномерного массива осуществляется через индекс (обычно целое число, являющееся порядковым номером элемента массива).

Пример описания одномерных массивов

```
Var
   A : Array [1..10] of Integer;
   B : Array [0..5] of String[7];
```

В данном примере описаны следующие переменные:

а) *A* – одномерный целый массив, состоящий из 10 элементов, первому элементу массива соответствует *A*[1], второму элементу – *A*[2], …, десятому элементу – *A*[10];

б) *B* – одномерный строковый массив, состоящий из шести элементов, первому элементу массива соответствует *B*[0], второму – *B*[1], …, шестому элементу массива – *B*[6].

Описание двумерного массива (матрицы)

```
Var
   <имя массива> : Array [Нач_инд1..Кон_инд1,Нач_инд2..Кон_инд2] of
<тип>;
```

где параметры `Нач_инд1`, `Кон_ инд1` указывают начальное и конечное значение номеров строк массива, параметры `Нач_инд2`, `Кон_ инд2` указывают начальное и конечное значение номеров столбцов массива.

Пример описания двумерных массивов

```
   Var
   C : Array [1..3, 1..5] of Real;
   D : Array [0..4, 1..2] of Byte;
```

В данном примере описаны следующие переменные:

а) *C* – двумерный вещественный массив, состоящий из трех строк (индексы строк от 1 до 3) и пяти столбцов (индексы столбцов от 1 до 5). Доступ к элементам массива *C* осуществляется так:

элемент первой строки и второго столбца массива – *C*[1, 2],

элемент третьей строки и четвертого столбца массива – *C*[3, 4];

б) *D* – двумерный байтовый массив, состоящий из пяти строк (индексы строк от 0 до 4) и двух столбцов (индексы столбцов от 1 до 2), доступ к элементам массива *D* осуществляется так:

элемент нулевой строки и второго столбца массива – *D*[0, 2],

элемент четвертой строки и первого столбца массива – *D*[4, 1].

П 1.5. Основные операторы языка Pascal

Оператор присваивания

```
<переменная> := <выражение>;
```

где `<переменная>` – переменная, элемент массива или имя функции, а `<выражение>` – любое допустимое выражение, совместимое по типу с переменной в левой части оператора. Оператор вычисляет значение выражения, записанного справа от операции присваивания «:=», и присваивает полученное значение переменной в левой части оператора.

Примеры использования оператора присваивания:

```
X := 10;          {присваивание переменной X значения 10}
P :=P + 5;        {увеличение значение переменной P на 5}
```

Операторы ввода-вывода

Для выполнения ввода-вывода данных используются операторы `Read`, `Readln`, `Write`, `Writeln`.

Структура оператора ввода

`Read(F, x1, x2, ..., xn)` или `Readln(F, x1, x2, ..., xn)`,

где `F` – переменная, связанная с файлом, из которого происходит чтение значений переменных `x1, x2, ..., xn`. Если переменная `F` отсутствует, то чтение значений переменных производится с клавиатуры. Оператор `Readln` в отличие от `Read` переводит курсор

на следующую строку. Использование оператора `Readln` без аргументов используется для ожидания нажатия клавиши <Enter>.

Структура оператора вывода:

`Write(F, e1, e2, ..., en)` или `Writeln(F, e1, e2, …, en)`, где `F` – переменная, связанная с файлом, в который осуществляется запись значений выражений `e1, e2, ..., en`. При отсутствии `F` значения выводятся на экран. Оператор `Writeln` в отличие от `Write` переводит курсор на новую строку после вывода значений. Использование оператора `Writeln` без аргументов используется для вывода пустой строки.

Выходные форматы языка Pascal

Для вывода значений переменных в языке Pascal используются следующие форматы (шаблоны):

$$R : a : b \text{ и } R : a,$$

где R – переменная;

a – число всех отводимых позиций под число;

b – число позиций дробной части числа.

Для целых чисел и строковых переменных используется формат $R : a$, для вещественных рекомендуется формат $R : a : b$, причем под знак и десятичную точку отводится отдельная позиция. Если для вещественного числа не указан выходной формат, то оно выводится в форме с плавающей точкой. Строки символов, заключенные в апострофы, выводятся как текст.

Примеры работы оператора `Write` представлены в табл. П 1.6. Если значение короче, оно «прижимается» к правому краю отведенного поля; если длиннее – поле «раздвигается» до необходимых размеров.

Таблица П 1.6

Значение	Оператор	Результат на экране
R = 654.734	Write(R)	6.54734000000171E+00002
R = 654.734	Write(R:10)	6.5E+0002
R = 654.734	Write(R:10:2)	654.73

Окончание табл. П 1.6

Значение	Оператор	Результат на экране
I = 77	Write(I)	77
I = 77	Write(I:5)	_ _ _77
S = Minsk	Write(S)	Minsk
S = Minsk	Write(S:3)	Min
S = Minsk	Write(S,' ',S)	Minsk Minsk

Оператор условного перехода

Условный оператор позволяет проверить некоторое условие и в зависимости от результата выполнить то или иное действие. Условный оператор обеспечивает ветвление вычислительного процесса. В языке Pascal используются два вида условных операторов.

1. **If** <условие> **then begin** <группа операторов 1> **end**
   else begin <группа операторов 2> **end;**
2. **If** <условие> **then begin** <группа операторов > **end;**

где `If, then, else` – служебные слова (если, то, иначе);

`begin ... end` – операторные скобки, ограничивающие действия операторов. Если используется один оператор, то операторные скобки могут быть опущены.

Для оператора первого вида если <условие> истинно, то выполняются <группа операторов 1>, если ложно, то <группа операторов 2>. Для оператора второго вида если <условие> истинно, то выполняется <группа операторов>. В общем случае <условие> представляет собой логическое выражение, в котором могут использоваться логические операции `not` («логическое отрицание»), `and` («логическое умножение И») и `or` («логическое сложение ИЛИ») и `xor` («логическое исключающее ИЛИ»).

Примеры использования оператора `if` приведены в табл. П 1.7.

Таблица П 1.7

Фрагмент программы	Математическая форма фрагмента
`If A <> B then Y:= sin(x);`	Если $A \neq B$, то вычислить $Y = \sin(x)$

`If (A > B) and (C < D) then`
`begin`
`  Y:= cos(x);`
`  Z := ln(abs(cos(x)));`
`end;`
	Если условия $A > B$ и $C < D$ выполняются одновременно, то вычислить $Y = \cos(x)$ и $Z = \ln(\lvert\cos(x)\rvert)$

`If (A + B) < D) then Y:= cos(x)` `else Y:= sin(x);`	Если условие $(A + B) < D$ выполняется, то вычислить $Y = \cos(x)$, иначе вычислить $Y = \sin(x)$

`If (R/D < D) or (A >0 ) then`
`begin`
`  Y:= ln(x);  Z := cos(x);`
`end else`
`begin`
`  Y:= sin(x)/cos(x);`
`  R := Y*Y;`
`end;`
	Если выполняется одно из условий: $R/D < D$ или $A > 0$, то вычислить $Y = \ln(x)$ и $Z = \cos(x)$, иначе вычислить $Y = \sin(x)/\cos(x)$ и $R = Y \cdot Y$

Операторы циклов

Операторы циклов используются для повтора фрагментов программы. В языке Pascal используется три вида операторов цикла: **for**, **While**, **Repeat**.

Структура оператора **for**

Движение цикла «вверх» (шаг +1)

```
for <n>:=<n1> to <n2>  do
begin
   <операторы>
end;
```

или

Движение цикла «вниз» (шаг –1)

```
for <n>:=<n1> downto <n2>  do
begin
   <операторы>
end;
```

где `n` – параметр цикла;

`n1, n2` – начальное и конечное значения параметра цикла;

`to, downto` – служебные слова, означающие движение цикла «вверх» и «вниз».

Пример использования оператора цикла for

Фрагмент программы реализации возведения переменной A в третью степень:

```
…
p:= 1;
for i:= 1 to 3 do p:= p * A;
…
```

Пошаговая иллюстрация работы оператора for

Шаг 1. $P = 1$;
Шаг 2. $I = 1$;
Шаг 3. Если $1 < 3$, то $P = 1 \cdot A$;
Шаг 4. $I = 1 + 1 = 2$;
Шаг 5. Если $2 < 3$, то $P = A \cdot A$;
Шаг 6. $I = 2 + 1 = 3$;
Шаг 7. Если $3 < 3$, то $P = A^2 \cdot A$;
Шаг 8. $I = 3 + 1 = 4$;
Шаг 9. Если $4 > 3$, то выход из цикла.

Цикл с предусловием – оператор While

```
While   <условие>   do
begin
<операторы>
end;
```

где `While, do` – служебные слова (пока, выполнить).

При выполнении `<условия>` выполняются `<операторы>`, после чего проверка `<условия>` повторяется. Если `<условие>` ложно (False), то «работа» **`While`** прекращается. Если тело цикла (`<операторы>`) состоит из одного оператора, то операторные скобки `begin …end` не используются.

Пример использования оператора `While`

Фрагмент программы реализации возведения переменной *A* в третью степень:

```
…
i:= 1; p:= 1;
While i <= 3 do
Begin
  p:= p * A;
  i:= i + 1;
End;
…
```

Цикл с постусловием – оператор Repeat

Repeat *<операторы>* **Until** *<условие>*,

где `Repeat`, `Until` – служебные слова (повторять до тех пор, пока).

Тело цикла (`<операторы>`) выполняется хотя бы один раз, после чего вычисляется `<условие>`, которое проверяется. Если его значение False, то `<операторы>` повторяются, в противном случае оператор `Repeat…Until` завершает свою работу.

Пример использования оператора `Repeat`

Фрагмент программы реализации возведения переменной *A* в третью степень:

```
…
i:= 1; p:= 1;
Repeat
  p:= p * A;
  i:= i + 1;
Until i > 3;
…
```

Сравнение циклов `While`, `Repeat`, `For`

В цикле `While` проверка условия выполнения цикла находится в начале цикла, а в `Repeat` – в конце. Цикл `Repeat` всегда выполняется

хотя бы один раз, а цикл `While` может не выполняться ни разу. В цикле `While` выход из цикла осуществляется, если условие ложно, а в `Repeat` – если условие истинно. При заранее известном числе циклов удобным является использование оператора `for`.

Операторы для работы с файлами

В языке Pascal существуют три класса файлов: типизированные, текстовые и нетипизированные. Каждому файлу в языке ставится в соответствие файловая переменная определенного типа. Для этого используется процедура

Assign*(f : File, name : String)*,

где `f` – переменная файлового типа;

name – строковое выражение, содержащее полное имя файла, включая диск и все подкаталоги.

При отсутствии диска и подкаталогов в выражении *name* файловая переменная будет ссылаться на файл, который находится или будет находиться в текущем каталоге на текущем диске. Данная процедура всегда предшествует другим процедурам работы с файлами, так как ставит в соответствие конкретному файлу на внешнем устройстве логическую файловую переменную языка, к которой впоследствии будут обращаться все другие файловые процедуры.

Для описания переменной текстового файла используется служебное слово `Text`. Например, запись `f:Text` означает описание файловой переменной `f`, которая будет использоваться для работы с текстовым файлом.

Для ввода и вывода информации в файл используются следующие процедуры:

Reset*(f : File)* – открывает существующий файл для чтения и записи;

Rewrite*(f : File)* – создает и открывает новый файл для записи.

В текстовом файле может быть использована процедура открытия файла с добавлением информации в конец файла:

AppEnd*(f : Text)*, где `f` – текстовая переменная.

Текстовый файл можно рассматривать как последовательность символов, разбитую на строки. Запись и чтение в файл осуществляются с помощью процедур ввода/вывода:

Read(*f, Ww :Word*) – чтение из файла, связанного переменной `f` и присваивание значений переменной `Ww`.

Write(*f, Ww :Word*) – осуществляет запись переменной `Ww` в файл, на который указывает файловая переменная `f`. Дополнительно можно использовать процедуры `Readln` и `Writeln`, которые обеспечивают те же действия, что и `Read` и `Write` соответственно, но при этом еще осуществляют перевод на новую строку.

Операция закрытия файла является логическим окончанием работы с любым открытым файлом. Для этого служит процедура

Close(*f : File*)

Использование процедуры `Close` позволяет устранить связь файловой переменной с внешним файлом, установленную с помощью процедуры `Assign`. После окончания работы с файлом обязательно требуется закрыть его.

Примеры ввода и вывода данных в файл

Разработать программу чтения трех чисел из файла *koef.txt* и вывод их произведения в файл *rez.txt*.

Структура файла *koef.txt*:

12.7 6.1 0.85

Решение

Шаг 1. Создать текстовый файл *koef.txt.*

Шаг 2. Записать в файл числа 12.7 6.1 0.85.

Шаг 3. Создать программу чтения чисел 12.7 6.1 0.85 и записи из произведения в файл *rez.txt*.

Шаг 4. Запустить программу.

Программа

```
Program ReadFile
Var
   {файловые переменные для работы с текстовыми файлами}
```

```
   f1, f2 : Text;
   k1, k2, k3 : Integer;
begin
   Assign(f1, 'koef.txt'); Reset(f1);{открытие файла koef.txt}
   Readln(f1, k1, k2, k3);{чтение значений из файла koef.txt}
   Close(f1);              {закрытие файла koef.txt}
   Assign(f2 'rez.txt'); Rewrite(f2); {создание файла rez.txt}
   {вывод значения произведения в файл rez.txt}
   Writeln(f2, 'Произведение коэффициентов = ', k1*k2*k3);
   Close(f2);        {закрытие файла rez.txt}
end.
```

П1.6. Использование стандартных процедур и функций модулей Crt и Graph в языке Pascal

В языке Pascal используются следующие варианты подпрограмм:

1) стандартные процедуры;
2) стандартные фунции;
3) процедуры пользователя;
4) функции пользователя.

Примеры стандартных процедур: TextMode, Close, Assign. Примеры стандартных функций: sin(*x*), cos(*x*), ln(*x*), exp(*x*). Стандартные процедуры и функции находятся в соответствующих библиотечных модулях языка. Так, например, процедуры и функции управлении экраном находятся в модуле Crt, процедуры и функции работы с графикой – в модуле Graph. Процедуры и функции пользователя могут включать в себя наборы стандартных процедур и функций.

Работа в текстовом режиме (модуль Crt)

Для использования процедур, которые находятся в модуле Crt необходимо использовать запись: **Uses** `Crt`.

Текстовые режимы отображают символы кодовой таблицы персонального компьютера. Минимальной единицей управления служит символ. Текстовые режимы устанавливаются процедурой `TextMode(`*режим*`)`. Значение *режим* – номер режима (если `режим = 0`, то производится установка экрана 25 строк × 40 столбцов, если `режим = 2`, то установка режима 25 строк × 80 столбцов). Например, в результате работы процедуры `TextMode(2)` будет установлен такой

текстовый режим, для которого координата *X* изменяется от 1 до 80, координата *Y* – от 1 до 25, координата левого верхнего угла (1, 1).

Процедуры управления экраном

Для управления экраном используются следующие процедуры:

ClrScr, которая очищает экран и помещает курсор в левый верхний угол экрана с координатами (1; 1);

GotoXY*(X, Y)*, помещающая курсор в точку экрана с координатами X (столбец) и Y (строка);

WhereX, возвращающая *X* координату текущей позиции курсора (результат целочисленный);

WhereY, возвращающая *Y* координату текущей позиции курсора (результат целочисленный).

Например, если курсор находится в 70-м столбце во 2-й строке, то результатом выполнения следующих операторов:

```
A := WhereX;
B := WhereY;
```

будет *A* = 70, *B* = 2.

Установка цвета и фона символов

Для установки цвета выводимых символов используется процедура TextColor (*цвет*), для фона – процедура TextBackGround(*цвет*). Значение *цвет* – это имя или номер константы цвета, возможные значения приведены в табл. П 1.8.

Таблица П 1.8

Номер константы	Имя константы	Цвет
0	Black	Черный
1	Blue	Синий
2	Green	Зеленый
3	Cyan	Голубой
4	Red	Красный
5	Magenta	Фиолетовый
6	Brown	Коричневый
7	LightGray	Светло-серый
…	…	…
14	Yellow	Желтый
15	White	Белый

Следующие примеры реализуют вывод строк с различным цветом и фоном:

```
TextColor(0);     {установка черного цвета символов}
TextBackGround(7); {установка светло-серого цвета фона}
Write('Черные символы на светлом фоне'); {вывод строки на экран}
TextColor(7);     {установка светло-серого цвета символов}
TextBackGround(0);      {установка черного цвета фона}
Write('Светлые символы на черном фоне'); {вывод строки на экран}
TextColor(Yellow); {установка желтого цвета символов}
TextBackGround(Red); {установка красного цвета фона}
Write('Желтые символы на красном фоне'); {вывод строки на экран}
```

Чтобы добавить при выводе эффект мерцания, при установке цвета символов указывается константа `Blink` (или 128), например:

```
TextColor(Green + Blink); { мерцающие зеленые символы}
TextColor(12 + 128);      {мерцающие светло-красные символы}
```

Управление звуком

Для управления частотой звука и его продолжительностью в языке Pascal используются стандартные процедуры `Sound`, `Nosound`, `Delay` из модуля `Crt`.

`Sound`*`(I)`* активизирует звуковые средства, где целочисленное значение `I` указывает частоту звучания звука в герцах в диапазоне 37–32767 Гц. Сила звука (громкость) не регулируется. Звук указанной частоты будет генерироваться до тех пор, пока не будет отменен процедурой `Nosound`.

`Nosound` – отмена звука. Отменяет звуковой режим, заданный процедурой `Sound`.

Для указания времени, в течение которого будет продолжаться звучание, используется процедура **`Delay`***`(I)`*. Значение *`I`* указывается в миллисекундах (мс).

Управление клавиатурой

Стандартная клавиатура имеет три типа клавиш:

– символьные (буквы, цифры);

– управляющие (функциональные, клавиши перемещения курсора, вставка и т. д.);

– сдвига (*Ctrl*, *Alt*, *NumLock*, *CapsLock*).

При нажатии клавиши микропроцессор клавиатуры вырабатывает код, который преобразуется системой Turbo Pascal в код сканирования. Символьные клавиши возвращают при нажатии одно значение (простой код), а управляющие – два, одно из которых ноль.

Управление клавиатурой осуществляется посредством специализированных функций `KeyPressed` и `ReadKey`. Функция `ReadKey` считывает символ с клавиатуры и возвращает значение типа `char` (табл. П 1.9). Чтение символа не сопровождается отображением его на экране. Функция `KeyPressed` возвращает булевое значение `True`, если была нажата какая-либо клавиша (табл. П 1.10), и `False` в противном случае.

Таблица П 1.9

Процедура	Клавиши		
	Символьные	Управляющие	сдвига
`KeyPressed`	`True`	`True`	`False`
`ReadKey`	Значение	#00+значение	–

Таблица П 1.10

Код	Значение	Код	Значение
1	2	3	4
71	Home	79	End
72	↑	80	↓
73	PgUp (страница вверх)	81	PgDn (страница вниз)
75	←	82	Ins (Вставка)
77	→	83	Del

Работа в графическом режиме (модуль Graph)

Модуль `Graph` предназначен для работы с графическими встроенными функциями, которые можно использовать в программе после подключения этого модуля с помощью зарезервированного слова `Uses`. Подключение осуществляется следующим образом:

```
Uses Graph;
```

Далее в программе следует загрузить графический драйвер для работы с монитором, например, для *egavga.bgi* – драйвер для EGAVGA-адаптера. Для этого выполняются две команды:

```
Uses Graph;
Var
   GraphDriver : Integer;
   GraphMode : Integer;
Begin
   GraphDriver : = Detect;
   InitGraph(GraphDriver,GraphMode, 'C:\PAS\BGI');
   CloseGraph;
End.
```

Первая команда присваивания устанавливает флаг для опознавания аппаратуры, а функция `InitGraph` автоматически обнаруживает аппаратные средства и загружает соответствующий графический драйвер, помещенный в каталоге «`C:\PAS\BGI`». Каталог может быть и другим и зависит от пути, где находится драйвер. Процедура `CloseGraph` останавливает работу графической системы и освобождает рабочие области, занятые графическим драйвером.

После подключения драйвера можно использовать встроенные функции и процедуры для работы в графическом режиме. Некоторые из них представлены ниже:

Arc(*x, y :Integer; нач_угол, кон_угол, радиус :Word)* – процедура вычерчивает дугу окружности с центром (`x, y`) и радиусом «*радиус*»; дуга рисуется от начального («*нач_угол*») до конечного угла (*«кон_угол»*);

Circle(*X, Y :Integer; радиус :Word)* – процедура вычерчивает окружность с центром (*X,Y*) и радиусом «*радиус*»;

Ellipse(`X, Y` *:Integer; нач_угол, кон_угол :Word; радX, радY : Word)* – процедура рисует эллиптическую дугу, где точка центра (`X,Y`), а «*радX*» и «*радY*» – горизонтальная и вертикальная оси; дуга вычерчивается от начального «*нач_угол*» до конечного угла «`кон_угол`»;

Line(*X1, Y1, X2, Y2 :Integer)* – процедура вычерчивает прямую линию из точки (*X1,Y1*) в точку (*X2,Y2*);

Lineto(*x, y :Integer)* – процедура вычерчивает прямую линию из точки, в которой находится текущий курсор, в точку с координатами (*x,y*);

Moveto*(x, y :Integer)* – процедура перемещает текущий указатель в точку с координатами (*x*, *y*);

OutText*(строка :String)* – процедура пересылает значение параметра «*строка*» на устройство вывода, начиная с текущего указателя;

OutTextXY*(X, Y :Integer;* текст_строка *:String)* – процедура выводит значение параметра «текст_строка» , начиная с точки, заданной координатами *(X,Y);*

Rectangle*(X1, Y1, X2, Y2 : Integer)* – процедура вычерчивает прямоугольник. *(X1,Y1)* – левый верхний угол, *(X2, Y2)* – правый нижний угол;

GetX *:Integer* – функция возвращает *X*-координату текущего указателя;

GetY *:Integer* – функция возвращает *Y*-координату текущего указателя;

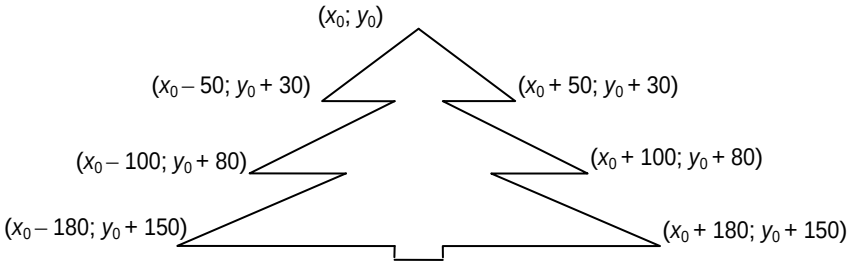
SetColor*(цвет :Word)* – процедура, используя палитру, устанавливает текущий цвет рисунка;

SetBkColor*(цвет :Word)* – процедура, используя палитру, устанавливает текущий цвет для фона;

SetLineStyle*(тип_строки :Word; образец :Word; толщина :Word)* – процедура устанавливает текущую толщину и тип линии;

SetTextStyle*(шрифт :Word; направление :Word; разм_символа : CharSizeType)* – процедура устанавливает текущий шрифт, тип и коэффициент размера символа.

Пример (*использование процедур модуля* Graph). Написать программу рисования «ели». Рисование «ели» производить поэтапно в цикле, до нажатия любой клавиши.



Программа

```
Program DemoGraph;
Uses Graph, Crt;
Var
   Gd, Gm    : Integer;
   X0, Y0, T : Integer;
Begin         {начало раздела операторов программы}
   Gd := Detect;
   InitGraph(Gd, Gm, 'E:\Pascal\BIN'); {открытие граф. режима}
   X0 := 300; Y0 := 100; {координаты начала рисования}
   T := 30000; {значение задержки времени для поэтапного рисования}
   Repeat   {начало цикла-выполнять, пока не нажата любая клавиши}
     {рисование ели с задержкой}
     MoveTo(X0, Y0);         LineTo(X0+50, Y0+30);  Delay(T);
     LineTo(X0+20, Y0+30);   LineTo(X0+100, Y0+80); Delay(T);
     LineTo(X0+50, Y0+80);   LineTo(X0+180, Y0+150);Delay(T);
     LineTo(X0+5, Y0+150);   LineTo(X0+5, Y0+155);  Delay(T);
     LineTo(X0-5, Y0+155);   LineTo(X0-5,Y0+150);   Delay(T);
     LineTo(X0-180, Y0+150);LineTo(X0-50, Y0+80);   Delay(T);
     LineTo(X0-100, Y0+80); LineTo(X0-20, Y0+30);   Delay(T);
     LineTo(X0-50, Y0+30);   LineTo(X0, Y0);        Delay(T);
     ClearDevice;          {очистка графического экрана}
   Until KeyPressed;       {конец цикла}
   CloseGraph;             {закрытие графического режима}
End.                       {конец программы}
```

Встроенные функции для обработки строк

Length*(str)* – функция вычисления длины строки *str* – типа *String*;

Delete*(str, i, j*) – функция удаления из строки *str* подстроки длиной *j* начиная с позиции *i*;

Insert*(str, str2, i)* – функция вставки строки str в строку *str2* с позиции i;

Copy*(str, i, j)* – функция выделения из *str* подстроки длиной *j* начиная с позиции *i*.

Пример (использование строковых переменных). Написать программу сортировки пяти фамилий Petrov, Sidorov, Pupkin, Turov, Ivanov по алфавиту.

Программа

```
Program DemoStr;
Uses Strings;
Var
   Stroks : array[1..10] of String;
   i, j, k : Integer;
   r : string;
Begin          {начало раздела операторов программы}
   {заполнение массива Stroks фамилиями}
   Stroks[1] := 'Petrov'; Stroks[2] := 'Sidorov';
   Stroks[3] := 'Pupkin'; Stroks[4] := 'Turov';
   Stroks[5] := 'Ivanov';
   for i := 4 downto 1 do
      for j :=1 to i do
      begin          {начало циклов 1, 2}
        k := 1;
        while k < Length(stroks[j]) do
        begin        {начало цикла 3 – побуквенное сравнение}
          if stroks[j,k] < stroks[j+1,k] then Break;
          if stroks[j,k] > stroks[j+1,k] then
          begin      {начало блока 1}
            {перемена местами строк stroks[j] и stroks[j+1]}
            r := stroks[j]; stroks[j] := stroks[j+1]; stroks[j+1] := r;
            Break;  {выход из цикла}
          end;       {конец блока 1}
          k := k + 1;
        end;         {конец циклов 1, 2}
      end;
   for i :=1 to 5 do writeln(Stroks[i]);  {вывод фамилий}
   Readln;           {ожидание нажатия клавиши Enter}
end.                 {конец программы}
```

Результат работы программы

```
Ivanov
Petrov
Pupkin
Sidorov
Turov
```

П 1.7. **Процедуры и функции пользователя**

Подпрограмма пользователя (процедура, ***функция***) – это независимая поименованная часть программы, предназначенная для выполнения определенных действий. После однократного предварительного описания подпрограмму при необходимости можно вызвать из любой позиции программы. По структуре ее можно рассматривать как мини-программу. Функция аналогична процедуре, но имеет два отличия:

1) функция передает в точку вызова скалярное значение (результат своей работы);

2) имя функции может входить в выражение как операнд, т. е. может использоваться в выражениях с различными вычислениями.

Описание подпрограммы

Описание подпрограммы включает заголовок (имя) и тело подпрограммы.

Заголовок процедуры состоит из зарезервированного слова `Procedure`, идентификатора (имени) процедуры и необязательного заключенного в круглые скобки списка формальных параметров с указанием типа каждого параметра.

```
Procedure  <имя> [(формальные параметры)];
Label …
Const …
Type …
Var
            < описание локальных переменных >
Begin
< раздел операторов >
End;
```

Заголовок функции содержит зарезервированное слово `Function`, идентификатор (имя) функции, заключенный в круглые скобки, необязательный список формальных параметров с указанием типа каждого параметра и тип возвращаемого функцией значения.

```
Function  <имя> [(формальные параметры)] : Тип результата;
Label …
```

```
Const …
Type …
Var
            < описание локальных переменных >
Begin
<раздел операторов>
End;
```

Например, `Function Prov(x, y, z : Integer) : Real;`

В разделе операторов должен находиться по крайней мере один оператор, присваивающий значение идентификатору (имени) функции.

Когда подпрограмма выполнит свои действия, программа продолжится с оператора, следующего непосредственно за оператором вызова процедуры.

Обращение к процедуре или функции производится с помощью оператора вызова:

```
Имя процедуры или функции (список фактических параметров);
```

Оператор вызова состоит из идентификатора (имени) процедуры или функции и списка фактических параметров, отделенных друг от друга запятыми и заключенных в скобки. Фактические параметры должны быть тех же типов и указаны в том же порядке, что и в описании подпрограммы.

Пример (программа с использованием процедуры без входных и выходных параметров). Написать программу с использованием процедуры, которая на центр экрана периодически (100 раз) выводит слово «Справочник».

Программа

```
Program DemoProc1;
Uses Crt;
Var i : Integer;
Procedure Ekran;              {заголовок процедуры}
Begin                         {начало процедуры Ekran}
   ClrScr;                    {очистка экрана}
   GotoXY (37, 11);           {курсор помещается в центр экрана}
   Writeln('Справочник');     {вывод на экран}
End;                          {конец процедуры Ekran}
```

```
Begin                          {начало основной программы}
   TextMode (2);               {установка текстового режима}
   for i:= 1 to 100 do Ekran;  {вызов процедуры Ekran 100 раз}
End.                           {конец основной программы}
```

Пример (программа с использованием двух процедур с входными параметрами). Написать программу с использованием процедур, которая в начале каждой строки экрана выводит слово «Справочник» с сопровождением звукового сигнала. Входными параметрами первой процедуры являются номера строки и столбца вывода строки. Входным параметром второй процедуры является частота звукового сигнала.

Программа

```
Program DemoProc2;
Uses Crt;
Var i : Integer;               {локальная переменная функции}
{Процедура Ekran
входные параметры: X, Y – номера строки и столбца}
Procedure Ekran(X, Y : Byte);  {заголовок процедуры}
Begin                          {начало процедуры Ekran}
   ClrScr;                     {очистка экрана}
   GotoXY (X, Y);{курсор помещается в точку с координатами X Y}
   Writeln('Справочник');      {вывод на экран слова 'Справочник'}
End;                           {конец процедуры Ekran}
{Процедура Zwuk
входной параметр: X – частота звукового сигнала}
Procedure Zwuk(X : Integer);   {заголовок процедуры}
Begin                          {начало процедуры Zwuk}
   Sound(X);                   {генерация звука частотой X}
   Delay(10);                  {задержка времени 10 мс}
   Nosound;                    {отмена звука}
End;                           {конец процедуры Zwuk}

Begin                          {начало основной программы}
   TextMode (2);               {установка текстового режима}
   for i:= 1 to 25 do
   begin
      Ekran(1, i); {вызов процедуры Ekran для каждой i-й строки экрана}
      {вызов процедуры Zwuk с увеличением частоты звукового сигнала}
      Zwuk(100 + 10*i);
   end;
End.                           {конец основной программы}
```

Пример (программа с использованием функции с входным и выходным параметрами). Составить программу с использованием функции для вычисления значения y по формуле

$$y = \begin{cases} \sin x, \text{ если } x > 0, \\ \cos x, \text{ если } x \le 0. \end{cases} \qquad (\text{П } 1.1)$$

Программа

```
Program DemoFunc1;
Var
   x1, y1 : Real;
{Функция F
входной параметр: x;
выходной параметр: значение, вычисленное по формуле (П 1.1)}
Function F(x : Real) : Real;      {заголовок функции}
Begin                             {начало функции F}
   {расчет значения функции в зависимости от величины x}
   If x > 0 Then F:= sin(x) Else F:= cos(x);
End;                              {конец функции F}
Begin                             {начало основной программы}
   Write('Введите x='); Readln(x1); {ввод значения x с клавиатуры}
   y1:= F(x1);              {вызов функции F для расчета}
   Writeln('Значение функции равно ', y1:5:2);{вывод результата}
End.                              {конец основной программы}
```

Пример (программа с использованием процедуры с входным и выходным параметрами). Составить программу с использованием процедуры для вычисления значения y по формуле

$$y = \begin{cases} \sin x, \text{ если } x > 0, \\ \cos x, \text{ если } x \le 0. \end{cases} \qquad (\text{П } 1.2)$$

Программа

```
Program DemoProc3;
Var
   x1, y1 : Real;
{Процедура F
входной параметр: x;
```

```
выходной параметр: y вычисленное по формуле (П 1.2)}
Procedure F(x : Real; Var y : Real)  {заголовок процедуры}
Begin                                {начало процедуры F}
   {вычисление переменной y в зависимости от величины x}
   If x > 0 Then y:= sin(x) Else y:= cos(x);
End;                                 {конец процедуры F}
Begin                                {начало основной программы}
   Write('Введите x='); Readln(x1); {ввод значения x с клавиатуры}
   F(x1, y1);                        {вызов процедуры F для расчета}
   Writeln('Значение функции равно ', y1:5:2);{вывод результата}
End.                                 {конец основной программы}
```

Пример (программа с использованием двух процедур, в которых в качестве входного и выходного параметров используется массив). Составить программу с использованием двух процедур, первая процедура формирует одномерный массив Y по закону, представленному ниже. Вторая процедура находит значение максимального элемента этого массива.

$$y_i = \begin{cases} \sin x_i, \text{ если } x > 0, \\ \cos x_i, \text{ если } x \le 0. \end{cases} \qquad (\text{П } 1.3)$$

Значения элементов x_i вводятся с клавиатуры, $i = 1, 10$.

Программа

```
Program Demo5;
Type
   TMas = array[1..10] of Real; {описание типа массива TMas}
Var          {описание глобальных переменных программы}
   X1, Y1 : TMas;
   MaxY1 : Real;
{Процедура FormY
входной параметр: X – массив;
выходной параметр: Y – массив, элементы которого вычисляют-
ся по формуле (П 1.3)}
Procedure FormY(X : TMas; Var Y : TMas);{заголовок процедуры}
Var    I : Byte; {описание локальных переменных процедуры}
Begin        {начало раздела операторов процедуры FormY}
   For I :=1 to 10 do
     If X[I] > 0 Then Y[I]:= sin(X[I]) Else Y:= cos(X[I]);
End;         {конец процедуры FormY}
```

```
{Процедура MaxArr
входной параметр: X – массив;
выходной параметр: MaxX – максимальный элемент массива X}
Procedure MaxArr(X : TMas; Var MaxX : Real);{заголовок процедуры}
Var    {раздел описания локальных переменных процедуры}
   I : Byte;
Begin        {начало раздела операторов процедуры MaxArr}
   MaxX := Y[1];
   For I :=2 to 10 do If Y[I] > MaxX Then MaxX:= Y[I];
End;         {конец процедуры MaxArr }
Begin        {начало раздела операторов основной программы}
   Write('Введите элементы массива X ');
   for I :=1 to 10 do   Read(X1[I]); {ввод значений с клавиатуры}
   FormY(X1, Y1);
   MaxArr(Y1, MaxY1) ;
   Writeln('Максимальный элемент массива Y равен ', MaxY1); {вывод}
End.         {конец основной программы}
```

Глобальные и локальные переменные

Глобальные переменные – это переменные, описанные в основной программе и доступные как программе, так и всем ее подпрограммам.

Локальные переменные – это переменные, описанные внутри подпрограммы и доступные только данной подпрограмме. Они могут быть описаны как в заголовке подпрограммы (формальные параметры), так и в разделе описания переменных.

Если переменная описана в основной программе (глобальная переменная) и не переопределена в подпрограмме, она может использоваться в подпрограмме и ее значение может измениться. Если эта переменная описана в подпрограмме (локальная переменная), то эти локальная и глобальная переменные между собой никак не связаны и при изменении значения локальной переменной в подпрограмме значение глобальной переменной в основной программе изменяться не будет.

Фактические и формальные параметры. Передача параметров в подпрограммы

Параметры подпрограммы – это значения, передаваемые в процедуру или функцию в качестве аргументов. При описании подпро-

граммы в скобках указываются имена переменных, содержащих передаваемые значения. Значение каждого фактического параметра при вызове процедуры передаётся формальному параметру. Существуют два способа передачи фактических параметров в подпрограмму: по значению и по ссылке.

С помощью первого способа (передача по значению) значение переменной фактического параметра при вызове подпрограммы присваивается локальной переменной (объявленной внутри подпрограммы), являющейся формальным параметром подпрограммы. Что бы потом ни происходило с локальной переменной, это никак не отразится на соответствующей глобальной (объявленной в основной программе).

Если же значение переменной необходимо изменить с помощью вычислений подпрограммы, то используют второй способ (передача по ссылке). При этом происходит следующее: при обращении к подпрограмме не происходит формирования локальной переменной формального параметра, а во время выполнения подпрограммы имя этой локальной переменной будет указывать на ту же область памяти, что и имя соответствующей глобальной переменной. Если в этом случае изменить локальную переменную, изменятся данные и в глобальной. Для использования этого способа передачи параметров перед именем переменной формального параметра в описании подпрограммы ставится служебное слово `Var`.

В нижеприведенных примерах направления стрелок указывают тип передачи фактического параметра в подпрограмму (по значению или по ссылке). Стрелка ↓ над параметром подпрограммы показывает, что подпрограмма только принимает значение этого параметра (передача по значению). Стрелка ↕ над параметром подпрограммы указывает, что подпрограмма принимает значение этого параметра и передает его в основную программу (передача по ссылке).

Примеры заголовков процедур

↕ ↕ ↓

`Procedure K1 (Var A, B : Integer; D : Real)` – процедура принимает значения переменных *A*, *B* по ссылке (значения меняются и

передаются в основную программу) и принимает значение переменной *D* по значению (просто принимает значение и не передает его).

Procedure K3 (R1, R2 : Integer; Var X1, X2 : Real) – процедура принимает значения переменных *R*1, *R*2 по значению (просто принимает и не передает их) и значения переменных *X*1, *X*2 по ссылке (значения меняются и передаются в основную программу).

Если в качестве передаваемого параметра используется массив, то заранее в разделе `Type` тип этого массива должен быть описан, а затем этот тип будет использоваться для описания как самого массива в основной программе, так описания в списке формальных параметров подпрограммы.

Например:

Procedure K2 (OK : Massiv) – передача в процедуру массива типа `Massiv`, который прежде должен быть описан:

```
Type Massiv = Array [1..12] of Real;
```

Пример (фрагмент программы с использованием процедуры)

```
Procedure Calc(X, Y, Z : Real; Var R1, R2 : Real);
Var U : Real;
begin
   U:= X*Y*Z;
   R1:= Cos(U); R2:= Sin(U);
end;
```

Эта процедура вызывается следующим образом:

```
Calc(2.1, 3.2, 8.5, Rez1, Rez2);
```

Пояснения к примеру. Формальные входные параметры *X*, *Y*, *Z* процедуры *Calc* принимают значения, соответствующие фактическим при вызове процедуры, а именно: *X* = 2.1, *Y* = 3.2, *Z* = 8.5. Используя эти значения, выполняется процедура. Результатами выполнения процедуры являются формальные параметры *R*1, *R*2, которые передают свои значения фактическим параметрам *Rez*1, *Rez*2.

Пример (программа с использованием процедур)

Программа

```
Program Demo7;
Var          {описание глобальных переменных программы}
   A1, A2 : Integer;
   J1, J2, J3 : Integer;
Procedure W1(R1, R2 : Integer;   {принимает R1, R2}
             Var X1, X2 : Integer);{X1, X2 – передает}
Begin                            {начало процедуры W1}
   X1:= R1 + R2;
   X2 := R1 * R2;
end;                             {конец процедуры W1}
Procedure W2(R3, R4 : Integer);  {принимает R3, R4}
Begin                            {начало процедуры W2}
   A1:= R3 - 4;  {A1, A2 – изменяются глобальные переменные}
   A2:= R4 + 4;
end;                             {конец процедуры W2}
   Procedure W3(Var R5, R6 : Integer);{передает R5, R6}
Begin                            {начало процедуры W3}
   R5:= J1 + 1;
   R6:= J2 + 7;
end;                             {конец процедуры W3}
begin                            {начало основной программы}
   J1:= 10; J2:= 30; J3:= 40;
   W1(J1, J2, A1, A2);           {вызов процедуры W1}
   {вывод результатов}
   Writeln(' J1 = ', J1,' J2 = ', J2,' A1 = ',A1, ' A2 = ', A2);
   W2(J2, J3);                   {вызов процедуры W2}
   Writeln('A1 = ', A1, ' A2 =  ', A2);{вывод результатов}
   W3(A1, A2);                   {вызов процедуры W3}
   {вывод результатов}
   Writeln(' J1 = ',J1,' J2 = ', J2,' A1 = ', A1,' A2 = ',A2);
   Readln;                      {ожидание нажатия клавиши Enter}
end.                             {конец основной программы}
```

Результаты работы программы:

```
J1 = 10 J2 = 30 A1 = 40 A2 = 300
A1 = 26 A2 =  44
J1 = 10 J2 = 30 A1 = 11 A2 = 37
```

Пояснения к примеру. При вызове процедуры *W*1 происходит следующее. Значения переменных *J*1, *J*2 передаются в процедуру (передача по значению), а значения переменных *A*1, *A*2 изменяются (передача по ссылке). Результаты выполнения процедуры – в первой строке.

При вызове процедуры *W*2 значения переменных *J*2, *J*3 передаются в процедуру (передача по значению), внутри процедуры изменяются значения глобальных переменных *A*1, *A*2. Результаты выполнения процедуры – во второй строке.

При вызове процедуры *W*3 значения переменных *A*1, *A*2 передаются и изменяются в процедуре (передача по ссылке), внутри процедуры используются глобальные переменные *J*1, *J*2. Результаты выполнения процедуры – третья строка.

Пример (программа с использованием процедуры). Написать программу вычисления $S = \sum_{I=1}^{3} \sum_{J=1}^{4} a_{ij}$, где a_{ij} – элементы матрицы *A*, формирующиеся по закону $a_{ij} = \frac{i}{j}$. Для вычисления суммы использовать процедуру.

Программа

```
Program Demo8;
Type
{описание типа Mas – двухмерного вещественного массива}
Mas = Array [1..3, 1..4] of Real;
Var                 {глобальные переменные программы}
   I, J : Byte;
   A : Mas;
   Result : Real;
Procedure Summa(B : Mas; Var C : Real);
Var I, J : Byte;           {локальные переменные процедуры}
Begin                      {начало процедуры Summa}
   C:= 0;
   for I:=1 to 3 do        {цикл по строкам матрицы B}
      for J:=1 to 4 do     {цикл по столбцам матрицы B}
         C:= C+ B[I, J];   {подсчет суммы элементов матрицы B}
end;                       {конец процедуры Summa}
Begin                      {начало основной программы}
   for I:=1 to 3 do        {цикл по строкам матрицы A}
     for J:=1 to 4 do      {цикл по столбцам матрицы A}
       A[I, J]:= I/J;      {вычисление элемента A[I, J] }
   Summa(A, Result);       {вызов процедуры Summa}
   {вывод результата}
   Writeln('Сумма элементов массива = ', Result:5:2);
   Readln;                 {ожидание нажатия клавиши Enter}
end.                       {конец основной программы}
```

Результаты работы программы

```
Сумма элементов массива =  12.50
```

Пояснения к примеру. При вызове процедуры *Summa* формальные параметры процедуры принимают значения фактических, а именно: массив *B* становится равным массиву *A*, значение *C* = *Result*. В процедуре производится суммирование элементов массива *B*, значение суммы содержится в переменной *C*. Значение *C* передается по ссылке, следовательно, при изменении *C* в процедуре также изменяется значение переменной *Result* в основной программе.

Пример (программа с использованием процедуры). Написать программу вычисления формирования массива *C* по закону $c_i = a_i \cdot b_i$, где a_i, b_i – элементы массивов *A* и *B*, $i = 1; 10$ (1). Для формирования элементов массива использовать процедуру, входными параметрами которой будут массивы *A* и *B*, выходной – массив *C*.

Программа

```
Program Demo9;
Type
   {описание типа Mas – одномерного вещественного массива}
   Mas = Array [1..10] of Real;
Var                          {глобальные переменные программы}
   I       : Byte;
   A, B    : Mas;
Procedure FormC(A, B : Mas; Var C : Mas);
Var I : Byte;                {локальная переменная процедуры}
Begin                        {начало процедуры FormC}
   for I:=1 to 10 do         {цикл по элементам массива C}
     C[I]:= A[I] * B[I];    {формирование C[I] элемента массива}
end;                         {конец процедуры FormC}
Begin                        {начало основной программы}
   for I:=1 to 10 do         {цикл по элементам массива A}
     Readln(A[I]);           {чтение I-го элемента массива A}
   for I:=1 to 10 do         {цикл по элементам массива B}
     Readln(B[I]);           {чтение I-го элемента массива B}
   FormC(A, B, C);           {вызов процедуры FormC}
   Readln;                   {ожидание нажатия клавиши Enter}
end.                         {конец основной программы}
```

П 1.8. Интегрированная среда Turbo Pascal

Главное меню содержит лишь оглавление дополнительных меню. В этих меню сгруппированы действия, близкие по своему роду, условное название которых и служит кодовым словом соответствующей опции главного меню:

File (файл) – действия с файлами и выход из системы;

Edit (редактировать) – восстановление испорченной строки и операции с временным буфером;

Search (искать) – поиск текста, процедуры, функции или места ошибки;

Run (выполнить) – выполнение программы;

Compile (компилировать) – компиляция программы;

Debug (отладка) – отладка программы;

Tools (инструменты) – вызов вспомогательных программ (утилит);

Options (опции) – установка параметров среды;

Window (окно) – работа с окнами;

Help (помощь) – обращение к справочной службе.

Рассмотрим некоторые пункты меню более подробно.

Меню *File* (файл)

New (новый)		Создание нового файла, по умолчанию этому файлу присваивается имя *NONAME.PAS*, которое можно изменить при записи файла на диск
Open… (открыть)	*<F3>*	Открытие в редакторе существующего текстового файла, в котором набрана либо программа, либо данные
Save (сохранить)	*<F2>*	Сохранение файла на диске; если имя файла дано по умолчанию, то пользователю предлагается изменить на другое имя, иначе сохраняется файл со старым именем
Save as… (сохранить как)		Сохранение файла с заданным пользователем именем
Change dir		Установка рабочего каталога
Exit (выход)	*<Alt+X>*	Выход из среды Turbo Pascal

Меню *Edit* (редактирование)

Undo (отменить)	*<Alt+BkSp>*	В активном окне редактора восстанавливает только что уничтоженную или измененную строку
Cut (вырезать)	*<Shift+Del>*	Удаляет (вырезает) выделенный блок из окна редактора и переносит его в буфер обмена
Copy (копировать)	*<Ctrl+Ins>*	Копирует выделенный блок из окна редактора в буфер обмена
Paste (вставить)	*<Shift+Ins>*	Вставляет содержимое буфера обмена в окно редактора с позиции курсора
Clear (очистить)	*<Ctrl+Del>*	Удаляет из окна редактора выделенный блок, но не помещает его в буфер обмена

Меню *Run* (выполнить)

Run (выполнить)	*<Ctrl+F9>*	Осуществляет компиляцию, компоновку и выполнение программы из окна редактора
Step Over	*<F8>*	Построчно выполняет программу (трассировка) без захода в процедуры и функции
Trace into	*<F7>*	Построчно выполняет программу (трассировка) с заходом в процедуры и функции
Go to cursor	*<F4>*	Выполняет программу до строки, где расположен курсор, а затем останавливается и передает управление встроенному отладчику
Program reset	*<Ctrl+F2>*	Прекращает выполнение программы

Меню *Compile* (компилировать)

Compile (компилировать)	*<Alt+F9>*	Компилирует программу в активном окне редактора

Меню *Debug* (отладк*а*)

Breakpoints…		Позволяет просмотреть все точки останова (контрольные точки) и при необходимости удалить, переместить любую точку или задать условия ее работы
Call stack	*<Ctrl+F3>*	Делает активным окно программного стека. В этом окне отображаются все вызовы процедур и функций

Watch		Делает активным окно наблюдения состояния переменных, добавленных командой Add Watch…
Output		Делает активным окно выходных результатов
User screen	<Alt+A5>	Делает активным окно пользователя и распахивает его во весь экран
Evaluate/modify…	<Ctrl+F4>	Дает возможность в процессе отладки просмотреть и изменить значение переменной или выражения
Add watch…	<Ctrl+F7>	Добавляет переменную в окно Watch
Add breakpoint…		Установка точки останова в текущей строке. Текущая строка – строка с курсором в окне редактора

П 1.9. Работа в интегрированной среде Turbo Pascal

Создание программы

1) Установить рабочий каталог (*File → Change dir*).

2) Создать новый файл (*File → New*). Откроется новое окно редактирования, по умолчанию имя файла *NONAME00.PAS*.

3) Сохранить программу на диске (*File → Save* или <*F2*>) или (*File → Save As…*).

4) В появившемся окне редактирования набрать программу или редактировать существующую.

5) Компилировать программу (*Compile → Compile* или <*Alt + F9*>). Если компиляция прошла успешно, то на экран выводится сообщение «*Compile successful: Press any key*» и для возврата в окно редактирования следует нажать любую кнопку. Если найдены ошибки, то компилятор укажет строку с ошибкой, в верхней строке будет ее описание. Ошибку следует исправить и выполнить компиляцию программы заново.

6) Выполнить программу (*Run → Run* или <*Ctrl + F9*>).

7) Посмотреть результат (*Debug → User Screen* или <*Alt + F5*>).

8) Выход из Паскаля (*File → Exit* или <*Alt + X*>).

Редактирование существующей программы

1) Открыть существующий файл (*File → Open* или <*F3*>).
2) В появившемся окне редактирования редактировать программу.
3) Далее – см. раздел «*Создание программы*», пункты 4–7.

Отладка программы

Отладка используется в том случае, если происходят ошибки или сбои при работе программы. Встроенный отладчик среды Turbo Pascal позволяет выполнять программу построчно, отдельными блоками, что дает возможность следить за изменениями значений переменных в ходе работы программы.

Производить отладку можно различными способами:

1) Пошаговое выполнение программы (трассировка). Для этого необходимо выбрать *Run → Step over* (<*F8*>) или *Run → Trace into* (<*F7*>). В первом случае подпрограмма выполняется за один шаг при трассировке, во втором – производится построчное выполнение подпрограммы. Начало выполнения – первый исполняемый оператор основной программы. После выполнения очередного оператора выполнение останавливается и у пользователя есть возможность провести анализ промежуточных результатов работы программы.

2) Выполнение программы до точки останова, для этого надо:

а) установить курсор на строке, на которой необходимо остановить выполнение;

б) установить точку останова – нажать на правую кнопку мыши и в локальном меню выбрать пункт *Toggle breakpoint* (<*Ctrl+F8*>) – эта строка подсветится красным цветом;

в) выполнить программу до точки останова, для этого выбрать *Run → Run* (<*Ctrl + F9*>). Выполнение программы остановится на выбранной строке (см. п.2, а, б), и далее можно продолжать отладку путем трассировки, анализируя результаты работы операторов;

г) убрать точки останова можно тем же способом, а именно, установить курсор на строке останова, в локальном меню выбрать *Toggle breakpoint* (<*Ctrl + F8*>), можно также это сделать с помощью пункта меню *Debug → Breakpoints*….

3) Выполнение программы, пока не станет истинным заданное условие, для этого надо:

а) установить курсор на строку, на которой необходимо остановить выполнение программы;

б) выбрать *Debug → Add Breakpoint…* и в появившемся диалоговом окне в поле *Condition* задать условие остановки (например, в цикле с параметром *i* = 1, 100 можно задать условие для одного из операторов цикла *i* = 10 – в этом случае выполнение остановится на этом операторе при *i* = 10).

4) Выполнение программы до позиции курсора, для этого надо:

а) установить курсор на строке, на которой необходимо остановить выполнение;

б) выбрать *Run → Go to cursor,* или в локальном меню пункт *Go to cursor*, или (<*F4*>).

Анализ промежуточных результатов работы операторов программы производится с помощью:

1) окна *Watch* – окно наблюдения состояния переменных и выражений; для работы с ним необходимо:

а) активизировать окно *Watch*, для этого выбрать *Debug → Watch* и внизу появится окно *Watch*;

б) задать переменные и/или выражения, значения которых необходимы для наблюдения, для этого необходимо выбрать *Debug → Add Watch…*, или в окне *Watch* дважды щелкнуть на пустой строке, или нажать <*Ins*> и в появившемся диалоговом окне ввести имя переменной или выражение;

в) указанные переменные и выражения вместе с их текущими значениями будут постоянно содержаться в окне наблюдения состояния переменных *Watch*;

2) окна *Evaluate/Modify*, которые позволяют просматривать текущие и задавать новые значения для переменных и выражений, для этого надо:

а) выбрать пункт меню *Debug → Evaluate/Modify* (<*Ctrl + F4*>);

б) в появившемся диалоговом окне в поле *Expression* задать имя переменной или выражение, после нажатия <*Enter*> или кнопки *Evaluate* в поле *Result* появится текущее значение заданной переменной, новое значение переменной необходимо задавать в поле *New value*.

Прервать выполнение программы во время отладки можно с помощью команды *Run → Program reset* (<*Ctrl + F2*>).

Пример отладки

Текст программы:

```
1.  Program DemoDebug;
2.  Var I, P  : Integer;
3.  R : Real;
4.  Begin
5.  for I:=-5 to 5 do
6.  Begin
7.     P := Sin(I)*Cos(I);
8.     R := 1/P;
9.     Writeln(R:5:2);
10. end;
11. end.
```

Алгоритм отладки программы

1) Запустить программу на компиляцию. Компиляция программы прошла успешно.

2) На этапе выполнения программы появится сообщение об ошибке «*Error 200. Division by zero*». Это означает, что произошло прерывание выполнения программы вследствие того, что знаменатель выражения стал равен нулю. Очевидно, что ошибка произошла при вычислении в седьмой строке. Необходимо найти, при каком значении *I* происходит ошибка, и исправить ее, для этого необходимо:

а) на восьмой строке программы поставить точку останова (щелкнуть мышью на этой строке и в локальном меню выбрать *Toggle breakpoint*);

б) активизировать окно *Watch* (пункт меню *Debug → Watch*) и задать в нем переменную для наблюдения – *P* (щелкнуть мышью в окне редактора, выбрать *Add Watch*...), для удобного отображения окон редактора и *Watch* можно воспользоваться пунктом меню *Window → Tile* – в результате окна не будут перекрываться;

в) запустить программу на выполнение (<*Ctrl + F9*>) – программа остановится на восьмой строке, в окне *Watch* будет отображено значение переменной *P*, затем продолжать выполнять программу (<*Ctrl + F9*>), параллельно отслеживая изменение переменной *P*; в результате будет обнаружено, что обнуление переменной *P* происходит при *I* = 0;

г) для того чтобы избавиться от ошибки, необходимо вставить проверку условия (P <> 0) в начале восьмой строки, в результате получим: `If p<>0 then R := 1/P.`

П 1.10. Правила и примеры построения схем алгоритмов

Алгоритм – совокупность правил и предписаний, выполнение которых приводит к решению поставленной задачи. Процесс построения алгоритма (конструирование) называется *алгоритмизацией*.

Правильно разработанный алгоритм обладает следующими свойствами:

1) *дискретность* – значения величин в каждый следующий момент времени должны получаться по определенным правилам из значений величин, имевшихся в предшествующий момент времени;

2) *определенность* (детерминированность) – каждое правило алгоритма должно быть однозначным, т. е. значения величин, получаемых в какой-то момент времени, однозначно связаны со значениями величин, вычисленных ранее;

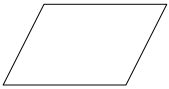
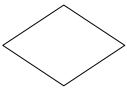
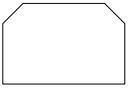
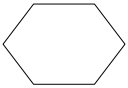
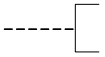
3) *результативность* (конечность) – алгоритм должен приводить к решению задачи за конечное число шагов;

4) *массовость* – алгоритм должен разрабатываться в общем виде так, чтобы его можно было применить для класса задач, различающихся лишь исходными данными.

Для оценки эффективности работы алгоритма используется характеристика вычислительной сложности алгоритма, которая может быть определена как сравнительное время решения задачи с помощью различных алгоритмов.

Для обеспечения хорошей наглядности и доступности при конструировании алгоритмов используется метод построения схем алгоритма. *Схемы алгоритмов* – графический способ записи последовательности выполнения некоторых процедур, изображаемых в виде графических символов определенной геометрической конфигурации, причем размеры этих символов и порядок построения схем регламентирован ГОСТ 19.701–90 (схемы алгоритмов, программ, данных и систем) (ИСО 5807–85). Наиболее часто употребляемые графические символы и соответствующие им процедуры приведены в табл. П 1.11.

Таблица П 1.11

Обозначение	Наименование
	Терминатор (отображает выход во внешнюю среду или вход из внешней среды)
	Данные (отображает данные, ввод-вывод данных)
	Процесс (отображает функцию обработки данных любого вид*а*)
	Решение (отображает решение или функцию переключательного типа, разветвление)
 	Граница цикла (начало … конец)
	Предопределенный процесс (подпрограмма, модуль)
	Соединитель (выход в часть схемы и вход из другой части этой схемы, используется для обрыва линии и продолжения ее в другом месте)
	Подготовка (отображает модификацию команды с целью воздействия на некоторую функцию, инициализация)
	Комментарий

Символы в схеме должны быть расположены равномерно и быть по возможности одного размера. Внутри символа необходимо помещать минимальное количество текста, необходимого для понимания функции данного символа. Если объем текста, помещаемого внутри символа, превышает его размер, следует использовать символ комментария. Потоки управления в схемах показываются линиями. Если направление потока отлично от стандартного (т. е. справа налево и снизу вверх), то стрелки должны указывать это направление.

Известно, что для реализации любого алгоритма достаточно трех базовых управляющих процессом обработки информации структур: композиция или следование, альтернатива или ветвление, итерация или цикл (теорема Бома–Якопини). Первая базовая структура «следование» реализуется линейным вычислительным алгоритмом. Второй базовой структурой является «ветвление», позволяющее в зависимости от результата проверки условия (истина или ложь) выбрать один из альтернативных путей работы алгоритма. Третья базовая структура «цикл» обеспечивает повторное выполнение операторов.

Примеры оформления алгоритмических структур согласно стандартам и соответствующие им фрагменты программы представлены в табл. П 1.12, П 1.13. Основными элементами программ являются структуры, реализующие ветвления и циклические структуры, и правила их оформления должны соответствовать ГОСТ 19.701–90. В частности, при построении схемы алгоритма для структуры, реализующей ветвление, используется символ «Решение», для циклических структур используется парный символ «Граница цикла» или символ «Подготовка». В табл. П 1.12 представлены различные элементы алгоритмических структур, реализующих ветвление. В табл. П 1.13 представлены различные элементы алгоритмических структур, реализующих циклы.

Таблица П 1.12

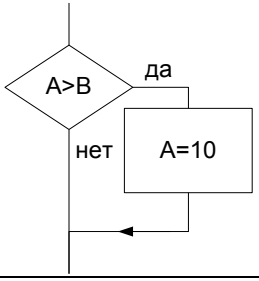
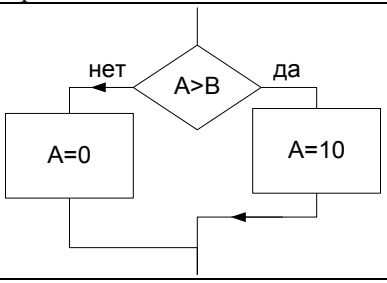
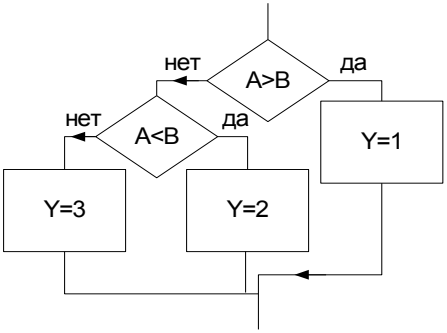
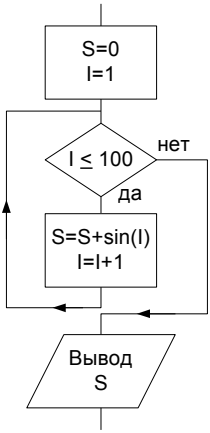
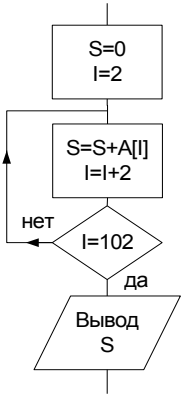
Описание задачи и фрагмент программы на языке Pascal	Элемент алгоритмической структуры
1	2
Пример 1	
Если $A > B$, то A =10	
If A > B **Then** A := 10;	
Пример 2	
$Y = \begin{cases} 10, \text{если } A > B; \\ 0, \text{если } A \le B \end{cases}$	
If A>B **Then** Y := 10 **else** Y := 0;	
Пример 3	
$Y = \begin{cases} 1, \text{если } A > B; \\ 2, \text{если } A < B; \\ 3, \text{если } A = B \end{cases}$	
If A>B **Then** Y := 1 **else If** A<B **Then** Y:= 2 **else** Y := 3;	

Таблица П 1.13

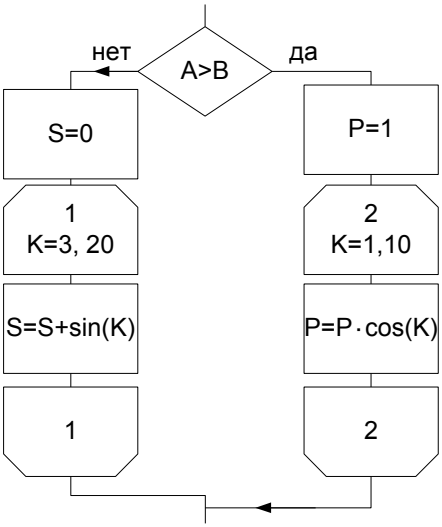
<table>
<tr><th>Описание задачи и фрагмент программы на языке Pascal</th><th>Элемент алгоритмической структуры</th></tr>
<tr><td>1</td><td>2</td></tr>
<tr><td colspan="2">Цикл с предусловием</td></tr>
<tr><td>$S=\sum_{i=1}^{100}\sin(i)$</td><td rowspan="2"> 
</td></tr>
<tr><td>

```
…
S := 0;
I := 1;
While I <= 100 do
Begin
     S := S + A[I];
     I := I + 1;
end;
Writeln(S);
…
```

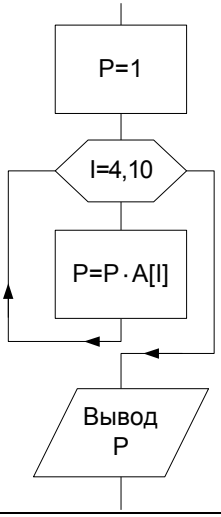
</td></tr>
<tr><td colspan="2">Цикл с постусловием</td></tr>
<tr><td>$S=\sum_{i=2}^{100}a_i$,
где A – множество четных чисел
$A=\{a_i\}=\{2,4,6,...,100\}$</td><td rowspan="2"> 
</td></tr>
<tr><td>

```
…
S := 0;
I := 2;
Repeat
     S:=S + A[I];
     I:=I + 2;
Until I = 102;
Writeln(S);
…
```

</td></tr>
</table>

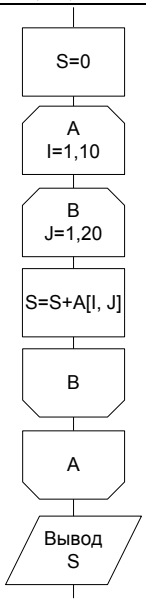
1	2
Цикл с параметром – с использованием парного символа «Граница цикла»	
$Y = \begin{cases} \prod_{k=1}^{10} \cos k, \text{ если } A > B \\ \sum_{k=3}^{20} \sin k, \text{ если } A \le B \end{cases}$	

`If A > B then`
`begin`
`  P := 1;`
`  for K := 1 to 10 do`
`  P := P * cos(K);`
`end else`
`  begin`
`  S := 0;`
`  for K := 3 to 20 do`
`  S:=S + sin(K);`
`  end;`	

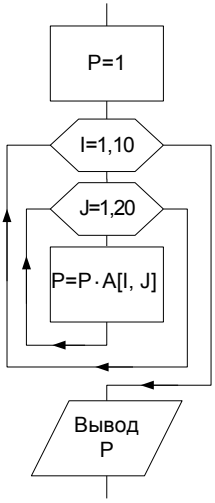
Цикл с параметром – с использованием символа «Подготовка»	
$P = \prod_{i=4}^{10} a_i$	

…
`P := 1;`
`for I :=4 to 10 do`
`  P := P * A[I];`
`Writeln(P);`
….	

Продолжение табл. П 1.13

1	2
Два цикла с параметрами – с использованием парных символов «Граница цикла»	
$S = \sum_{i=1}^{10} \sum_{j=1}^{20} a_{ij}$	

`S := 0;`
`for I :=1 to 10 do`
`  for J :=1 to 20 do`
`  S := S + A[I, J];`
`Writeln(S);`
…	

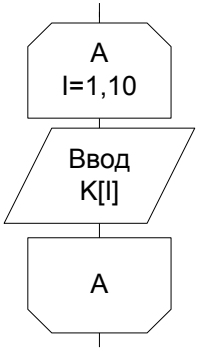
Два цикла с параметрами – с использованием символа «Подготовка»	
$P = \prod_{i=1}^{10} \prod_{j=1}^{20} a_{ij}$	

…
`P := 1;`
`for I :=1 to 10 do`
`  for J :=1 to 20 do`
`  P := P * A[I, J];`
`Writeln(P);`
…	

Продолжение табл. П 1.13

<table>
<tr><th>1</th><th>2</th></tr>
<tr><td colspan="2">Ввод одномерного массива K_{10} –
с использованием парного символа «Граница цикла»</td></tr>
<tr><td>

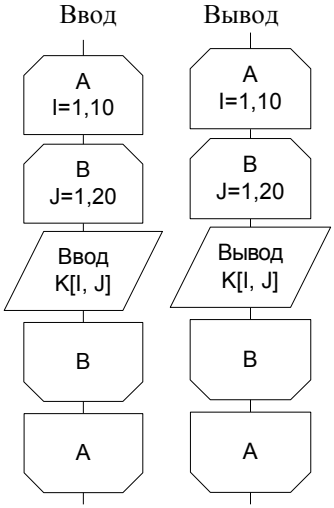
```
…
for I :=1 to 10 do
   Read(K[I]);
…
```

</td><td> 
</td></tr>
<tr><td colspan="2">Ввод (вывод) двумерного массива $K_{10\times20}$ –
с использованием парных символов «Граница цикла»</td></tr>
<tr><td>Построчный ввод

```
…
for I :=1 to 10 do
begin
   for J :=1 to 20 do
      Read(K[I, J]);
   Readln;
end;
…
```

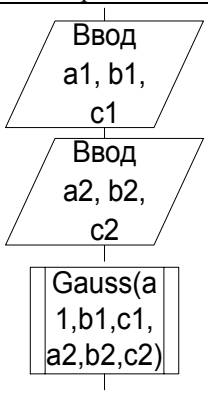
Построчный вывод

```
…
for I :=1 to 10 do
begin
   for J :=1 to 20 do
      Write(K[I, J], ' ');
   Writeln;
end;
…
```

</td><td> 
</td></tr>
</table>

Окончание табл. П 1.13

1	2
Использование символа предопределенного процесса	

Фрагмент программы
с использованием процедуры *Gauss*
…
`Readln(a1, b1, c1);`
`Readln(a2, b2, c2);`
`Gauss(a1, b1, c1, a2, b2, c2, x1, x2);`
…
	

П 1.11. Представление схем алгоритмов с помощью Microsoft Word

Чаще всего для построения схем алгоритмов используют инструменты Microsoft Word. В данном подразделе будет приведено описание основных операций, используемых для создания схем, в среде Microsoft Word.

Создание символов производится с помощью инструментов пункта *Автофигуры* → *Блок схема* (рис. П 1.1) панели *Рисование* (рис. П 1.2). Включить отображение панели *Рисование* можно с помощью кнопки *Панель рисования*  на панели *Стандартная* или выбором пункта меню *Вид* → *Панели Инструментов* → *Панель рисования*. Так, например, организация цикла производится с помощью автофигуры «Подготовка»  .

Добавление текста необходимо производить с помощью пункта *Добавить текст* локального меню автофигуры (появляется после щелчка правой кнопкой на автофигуре). Если текст не помещается в символе, то следует написать его в комментарии к этому символу, сконструировав его из линий.

Выделение группы символов производится с помощью клавиши <*Shift*>.

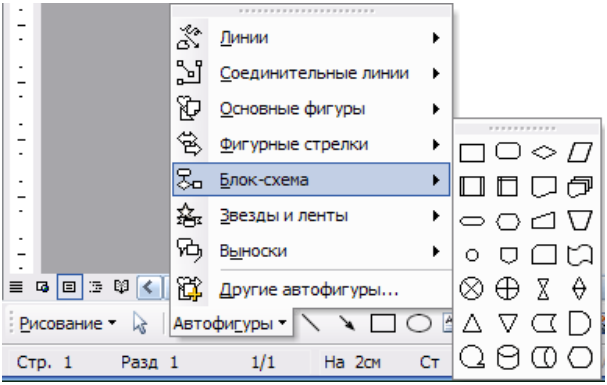


Рис. П 1.1

Рис. П 1.2

Указание размеров производится следующим образом: необходимо выделить символ или группу символов и дважды щелкнуть на выделенном символе или выбрать пункт меню *Формат* → *Автофигура*, затем перейти на вкладку *Размер* и в полях *Ширина* и *Высота* указать необходимые значения.

Выравнивание символов производится следующим образом:

1) выделить необходимые символы с помощью инструмента *Выбор объектов*  или последовательно выделяя их щелчком мыши с одновременно нажатой клавишей <*Shift*>;

2) активизировать команду *Действия* на панели *Рисование*: *Действия* → *Выровнять/распределить*;

3) из открывшегося списка выбрать необходимую команду для работы с выделенными объектами.

Необходимо обратить внимание на наличие флажка у пункта *Относительно страницы* (*Действия* → *Выровнять/распределить* → *Относительно страницы*). Если он отсутствует, то выров-

нивание символов производится относительно друг друга, а если он имеется – то относительно страницы.

Соединение символов производится с помощью инструментов пункта *Автофигуры* → *Соединительные линии* (рис. П 1.3). Инструмент *Соединительные линии* работает только на полотне рисунка. Выбрав соединительную линию, надо указать мышью места закрепления соединительной линии. При перемещении указателя мыши по символам места возможных соединений отображаются в виде синих кружков. Следует подвести указатель к первому символу и щелкнуть на месте начала соединения, затем подвести указатель мыши на другую фигуру и щелкнуть на месте конца соединения.

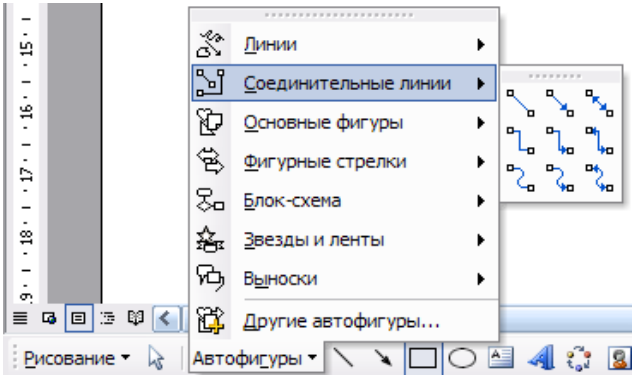


Рис. П1.3

Указание направления действия в алгоритме производится в том случае, если оно отлично от стандартного, а именно – справа налево и снизу вверх. Для этого необходимо использовать инструмент *Стрелка* на панели *Рисование*. Если необходимо указать на рисунке несколько стрелок, то лучше копировать уже имеющиеся, поскольку при этом не происходит изменения направления соединительных линий.

Для более точного подбора положений символов и стрелок можно использовать одновременное нажатие клавиши <*Ctrl*> и клавиш движения курсора, а для подбора положения линии можно использовать одновременное нажатие клавиши <*Alt*> и кнопки мыши.

Вставка слов «да» и «нет» возле символа ветвления производится с помощью инструмента *Надпись*  на панели *Рисование*. Надпись следует отформатировать, убрав заливку и рамку: *Формат* → *Надпись*→ вкладка *Цвета и линии*. Для поля *цвет Заливки* выбрать значение *«Нет заливки»,* для поля *цвет линии* – значение *«Нет линий»*.

Добавление комментария к схеме алгоритма производится следующим образом:

а) добавить символ *Автофигуры* → *Основные фигуры* → *Левая круглая скобка* (рис. П 1.4);

б) желтый маркер добавленной автофигуры переместить в крайнее левое верхнее положение, для того чтобы скобка приобрела вид квадратной скобки;

в) соединить соответствующий символ и скобку соединительной линией, сделав ее пунктирной с помощью инструмента *Тип штриха*;

г) вставить текст комментария с помощью инструмента *Надпись*.

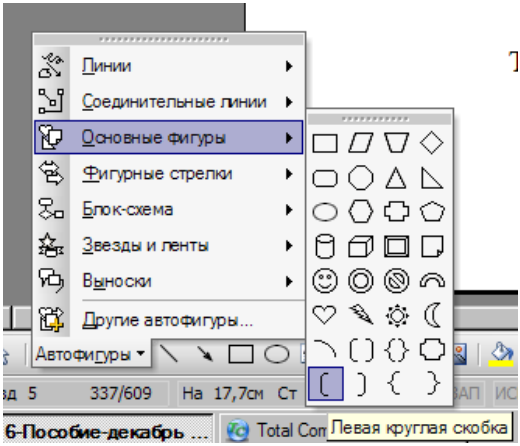
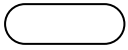


Рис. П 1.4

Пример пошагового построения схемы алгоритма

Используя инструментарий MS Word, необходимо построить схему алгоритма для табуляции функции $y = \log_3(x^3 - 4) + \sqrt{\sin^2 x}$ для $x = -10, 100\ (0{,}1)$.

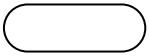
Шаг 1. Определить типов символов исходя из анализа условия задачи:

а) символ начала алгоритма – терминатор  ;

б) для перебора значений *x* необходимо организовать цикл с помощью символа *Решение*  , с его помощью организуется проверка окончания цикла;

в) для вычисления значения *y* необходимо использовать символ *Процесс*  ;

г) для вывода значений на экран необходимо использовать символ *Данные*  ;

д) символ окончания алгоритма – терминатор  .

Шаг 2. С помощью команды панель *Рисования* → *Автофигуры* → *Блок-схема* добавить символы в схему алгоритма в определенной последовательности один под другим.

Шаг 3. Задать необходимые размеры символов с помощью команды *Формат* → *Автофигуры* → *Размер* → *Ширина* и *Высота*, а именно: для поля *Ширина* задать значение 3 см, для поля *Высота* – 2 см, для символов начала и конца алгоритма для поля *Высота* – 1 см.

Шаг 4. Выровнять символы, размещенные по центру:

а) установить флажок у пункта *Относительно страницы* (*Действия* → *Выровнять/распределить* → *Относительно страницы*);

б) выделить группу символов с помощью клавиши *Shift* или инструмента *Выбор объектов*;

в) применить команду *Действия* → *Выровнять/распределить* → *Выровнять по центру*.

Шаг 5. Выровнять символы, размещенные справа (символы *Процесс* и *Данные*):

а) убрать флажок у пункта *Относительно страницы* (*Действия* → *Выровнять/распределить* → *Относительно страницы*);

б) выделить группу символов с помощью клавиши <*Shift*> или инструмента *Выбор объектов*;

в) применить команду *Выровнять по центру* к выделенным символам.

Шаг 6. Добавить в символы текст. Для этого необходимо щелкнуть на символе, в локальном меню выбрать команду *Добавить*. Так, например, в символ начала алгоритма необходимо добавить текст «Начало», в следующий символ *Процесс* «*X* = –10». Отформатировать текст таким образом, чтобы он располагался по центру символа.

Результаты *шагов* 1–6 представлены на рис. П 1.5.

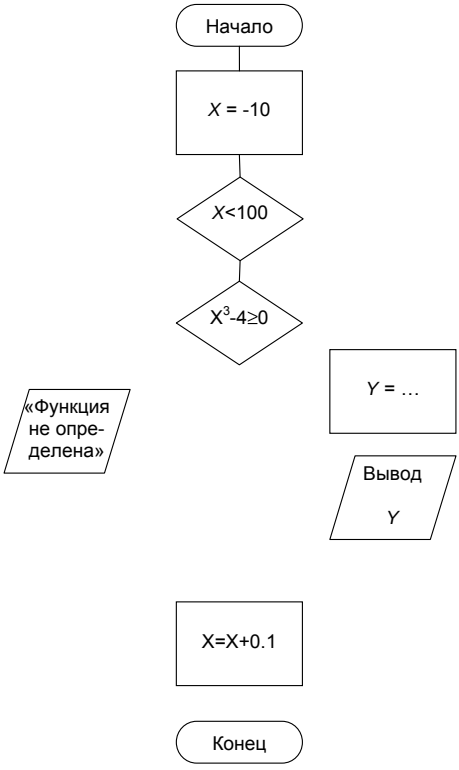


Рис. П 1.5

Шаг 7. Поскольку весь необходимый текст не помещается в символе процесса, необходимо добавить комментарий:

а) вставить скобку с помощью инструмента *Автофигуры* → → *Основные фигуры* → *Левая круглая скобка*;

б) соединить её с символом схемы пунктирной соединительной линией;

в) вставить в комментарий текст с помощью инструмента *Надпись* (в данном случае создается формула в редакторе формул *Microsoft Equation*).

Шаг 8. Вставить надписи «*да*» и «*нет*» возле символов, реализующих ветвление с помощью инструмента *Надпись* панели *Рисование*.

Шаг 9. Соединить символы соединительными линиями с использованием инструментов панели *Рисование: Автофигуры* → *Соединительные линии*. Указать направление действия алгоритма (где это необходимо) стрелками с помощью инструмента *Стрелка* на панели *Рисование*.

Результаты *шагов 7–9* представлены на рис. П 1.6.

Результаты *шагов 7–9* представлены на рис. П 1.7.

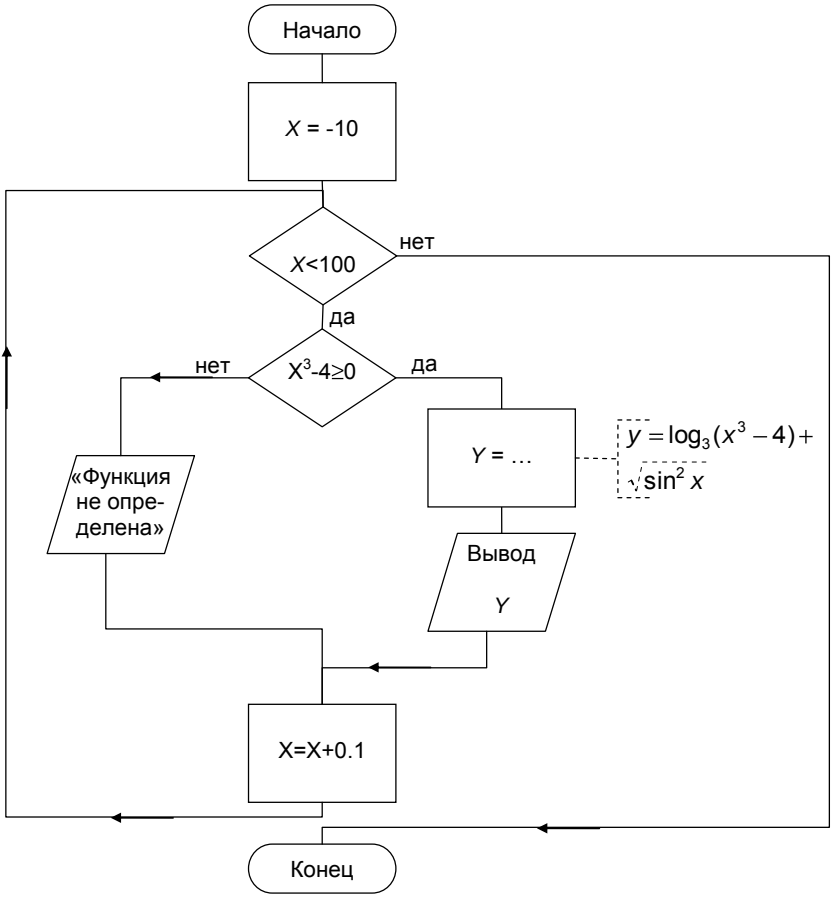


Рис. П 1.7
Рис. П 1.6

П 1.12. Представление схем алгоритмов с помощью программы Microsoft Visio

Для построения схем алгоритмов можно использовать инструменты программы Microsoft Visio (исполнение рисунков данного подраздела проводилось в Microsoft Visio 2003).

Последовательность действий для построения схемы алгоритмов:

Шаг 1. Определить типы символов, исходя из анализа условия задачи.

Шаг 2. Создать символы путем перемещения объектов на страницу документа из вкладки *Basic Flowchart Shapes* и расположить их в необходимом порядке следования. Размеры символов, координаты и угол расположения можно задавать с помощью панели *Размер и положение*, активизация которой производится с помощью выбора пункта *Вид* → *Размер и положение*.

Шаг 3. Ввести в символы необходимый текст (активизируется двойным щелчком мыши) или формулу (выбор пункта *Вставка* → *Объект…* → *Microsoft Equation*). Текст можно вращать с помощью кнопки *Текстовый блок* (рис. П 1.7).

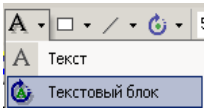


Рис. П 1.7

Шаг 4. Соединить символы с помощью инструмента *Соединитель* (кнопка *Соединитель* на панели *Стандартная*) и точек соединения (рис. П 1.8): активизировав кнопку *Соединитель*, щелкнуть по точке соединения первого символа и появившуюся линию протянуть, не отпуская мыши, до точки соединения со вторым символом. После соединения символов щелкнуть по кнопке *Указатель* (активизация кнопки *Указатель* позволяет завершить работу с объектами определенных групп).

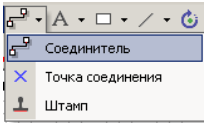


Рис. П. 1.8

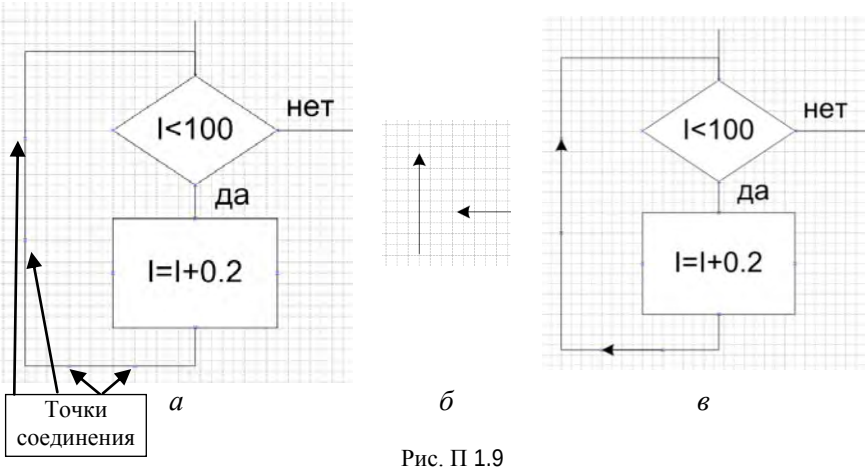
Можно использовать как уже имеющиеся точки соединения символов, так и создавать новые. Для этого необходимо выделить щелчком символ или линию →, щелкнуть по кнопке *Точка соединения* → щелкнуть в месте расположения новой точки соединения при нажатой клавише <*Ctrl*> (появится синий крестик, который будет перемещаться вместе с символом). Такой способ соединения символов удобен тем, что при перемещении символов линии соединения не обрываются.

Шаг 5. Нарисовать стрелки:

1) вставить на линии по две точки соединения (рис. П 1.9, *а*);

2) создать стрелку с помощью инструмента *Линия* на панели *Стандартная*  и кнопки *Концы линий* панели *Форматирование*  (рис. П 1.9, *б*);

3) вставить стрелки в точки соединения (рис. П 1.9, *в*), для чего последовательно подтягивать концы стрелок к соответствующим местам соединения.



а *б* *в*

Рис. П 1.9

Шаг 6. Вставить подписи «*да*», «*нет*» для символов решения с помощью кнопки *Текст* на панели *Стандартная* (см. рис. П 1.7).

Шаг 7. Пронумеровать символы с помощью инструмента *Текст* (см. рис. П 1.7).

Шаг 8. Выровнять символы:

а) выделить объекты;

б) выбрать пункт меню *Форма → Выровнять формы*;

в) в диалоговом окне *Выровнять формы* задать параметры выравнивания.

Шаг 9. Сгруппировать символы:

а) выделить объекты;

б) выбрать пункт *Группировать* в основном или локальном меню *Форма*.

П 1.13. **Пояснения к решению некоторых задач пособия**

Табулирование (табуляция) функции

Табулирование (табуляция) *функции* – это составление таблицы значений функции, т. е. для определенных значений переменных или переменной находятся соответствующие значения функции.

При программировании реализация табулирования предполагает организацию цикла или циклов (если используется несколько переменных) для перебора значений переменных и вычисление функции.

Перебор значений переменной можно проводить различными способами:

1) если переменная x представлена в виде массива чисел $X[N]$, то для перебора можно использовать цикл *for…to…do* с параметром цикла $I = 1, \ldots, N$, где N – размер (число элементов) массива;

2) если переменная x изменяется по закону $x = x_n$, x_k (h), где x_n – начальное значение переменной x; x_k – конечное значение переменной x; h – шаг изменения, то для перебора значений x можно использовать циклы *While… d*o или *Repeat … Until*. В этом случае необходимо:

а) до цикла задать начальное значение $x = x_n$,

б) в конце тела цикла произвести увеличение x на шаг изменения: $x = x + h$;

в) в качестве условия окончания перебора значений для цикла *While…do* использовать условие $x \le x_k$ (пока $x \le x_k$, выполнять операторы цикла), для цикла *Repeat…Until* – условие $x > x_k$ (выполнять операторы цикла до тех пор, пока x не станет больше x_k).

Например, пусть переменная x изменяется по закону $x = 2{,}5; 10\ (0{,}1)$, т. е. $x_n = 2{,}5$, $x_k = 10$, $h = 0{,}1$, тогда циклы перебора переменных следует организовать следующим образом:

Фрагмент программы с использованием цикла *While…do*	Фрагмент программы с использованием цикла *Repeat … Until*

…
X := 2.5;
While X <= 10 **do**
Begin
<операторы цикла>
X := X + 0,1;
End;
…
	…
	X := 2.5;
	Repeat
	<операторы цикла>
	X := X + 0,1;
	Until X > 10;
	…

3) Для перебора значений *x*, заданных по закону $x = x_n, x_k (h)$, можно также использовать цикл *for…to…do*. В этом случае необходимо:

а) до цикла определить число итераций цикла, например, с помощью оператора: *IMax:= Trunc((Xk – Xn)/Xh)+1*; функция *Trunc*() – возвращает значение целой части аргумента;

б) в цикле значение *x* для перебора определяется с помощью оператора *X := Xn + (I – 1)*h*.

Например, пусть переменная *x* изменяется от 1 до 6 с шагом 2 (т. е. может принимать значения 1, 3, 5). Таким образом, $x_n = 1$, $x_k = 6$, $h = 2$, а число итераций цикла будет равно трем. Цикл следует организовать следующим образом:

Фрагмент программы	Пояснения

…
IMax:=Trunc((6-1)/2)+1;
for I:=1 **to** IMax **do**
Begin
X:= Xn + (I - 1)*Xh;
<операторы цикла>
end;
…
	IMax:= *Trunc*(2.5)+1=2+1=3;
	1-я итерация: $I = 1$, $X = 1 + (1 – 1)\cdot 2 = 1$;
	2-я итерация: $I = 2$, $X = 1 + (2 – 1)\cdot 2 = 3$;
	3-я итерация: $I = 3$, $X = 1 + (3 – 1)\cdot 2 = 5$.

Формирование массива по заданному закону

1) Организация одного цикла (для одномерного массив*а*) или двух циклов (для двумерного массива) *for…to… do* для перебора всех значений массива. Максимально возможные параметры цикла (в примере это *I*, *J*) соответствуют размеру массива, например, если

массив описывается как $A_{20\times30}$, т. е. состоит из 20 строк и 30 столбцов, то $I = 1, 2, \ldots, 20$ и $J = 1, 2, \ldots, 30$.

2) В теле цикла осуществляется присваивание текущему элементу массива $A[I, J]$ определенного значения. Если для формирования массива указано одно или несколько условий, то с помощью оператора или операторов *If... Then* организуется ветвление.

Пример

```
…
Var A : Array [1..20, 1..30] of Real;
…
for I:=1 to 20 do
  for J:=1 to 30 do
    If I=J Then A[I, J]  := …
           Else A[I, J]:=…
…
```

3) Если в цикле *for...to...do* или в операторе *If...Then* только один оператор (даже если он занимает несколько строк), то операторные скобки *Begin...end* можно не ставить.

Вывод двумерного массива (матрицы) на экран или в файл

1) Организация двух циклов (для двумерного массива) *for...to... do* для перебора всех значений массива.

2) Второй вложенный цикл будет содержать оператор *Write*(*A*[*I*, *J*]) для вывода через пробелы значения элементов массива *J*-го столбца. После этого цикла в конце первого цикла необходимо использовать оператор *Writeln*() – переход на другую строку, т. е. сначала произойдет вывод первой строки массива, потом переход на другую строку, затем вывод второй строки, переход на следующую строку и т. д.

Пример

```
…
Var A : Array [1..20, 1..30] of Real;
…
for I:=1 to 20 do
Begin
  for J:=1 to 30 do Write(A[I, J]);
  Writeln;
```

```
end;
…
```

Поиск минимального элемента массива и его номера

1) Произвести в разделе *Var* описание массива, переменной для минимального значения массива того же типа, что и элементы массива (в примере это переменная *Min*), переменной для номера элемента массива (*IMin*) в случае одномерного массива или переменных для номеров строки и столбца минимального значения (*IMin*, *JMin*) в случае двумерного массива. Например:

```
…
Var
  A : Array [1..20, 1..30] of Real;
  Min : Real;
  IMin : Byte;
…
for I:=1 to 20 do
  for J:=1 to 30 do
    If A[I, J] < Min Then
Begin
  Min  := A[I, J];
  IMin := I;
end;
…
```

2) Присвоить переменной минимума (*Min*) значение выбранного элемента массива, например *A*[1], а переменной номера минимального элемента (*IMin*) – соответствующее значение, например 1.

3) Организовать один цикл (для одномерного массива) или два цикла (для двумерного массив*а*) *for…to… do* для перебора всех необходимых значений массива. Параметры цикла (в примере это *I*, *J*) соответствуют размеру массива. Для одномерного массива начальное значение параметра цикла равно 2 (поскольку значение первого элемента присвоено переменной для минимума).

4) В теле цикла производится сравнение переменной минимума (*Min*) с очередным значением элемента массива *A*[*I*]. Если значение элемента массива *A*[*I*] меньше, то переменной минимума присваивается значение *A*[*I*], а переменной номера (*IMin*) присваивается значение *I*, если значение элемента массива не больше, то все остается без изменений.

Поиск максимального элемента массива и его номера

Действие производится аналогично поиску минимального элемента, отличие состоит в том, что если при сравнении значение очередного элемента массива *A*[*I*] больше максимума (*Max*), то переменной максимума присваивается значение элемента массива.

Организация табличного вывода результатов

1) Вывести на экран шапку таблицы (горизонтальные линии, с помощью символов «-» или «_» и названия столбцов), длина горизонтальной линии и расположение шапки подбираются.

2) При выводе в цикле значений использовать символ «|» – разделение столбцов и форматный вывод (для того чтобы столбцы получались одинаковой ширины).

3) После цикла вывести горизонтальную линию.

Нахождение суммы бесконечного числа элементов ряда

Поиск суммы бесконечного числа элементов ряда производится с помощью циклов *While…do* или *Repeat…Until*. Условие окончания суммирования – разность предыдущего и текущего членов ряда – должна быть меньше определенной величины (погрешности). Ниже приведено описание действий для цикла *Repeat…Until*.

1) Описать в разделе *Var*:

а) две переменные, в которых будут храниться текущий и следующий элементы ряда (*S*1, *S*2);

б) переменную для вычисления членов ряда (*X*);

в) переменную суммы ряда (*S*).

2) Присвоить переменной *S*2 – первый элемент ряда – начальное значение (значение 1), переменной суммы ряда – значение первого элемента ряда (*S*2).

3) Для суммирования членов ряда организуется *Repeat …Until*. В теле цикла выполняются действия:

а) переменной *S*1 (значение предыдущего члена) присвоить значение *S*2, т. е. значение текущего элемента ряда на следующей итерации становится значением предыдущего элемента;

б) вычислить следующий член ряда (переменная *S*2);

в) к сумме ряда (*S*) прибавить следующий член ряда (*S*2);

г) проверить условие продолжения суммирования, а именно: разность текущего и следующего элементов ряда не должна быть меньше определенной величины (погрешности): если условие выполняется, то продолжается выполнение цикла (пункты 3, *а–в*), если нет – выход из цикла.

ПРИЛОЖЕНИЕ 2

П 2.1. Основные теоретические сведения о численных методах решения систем линейных уравнений

П 2.1.1. Матричный метод

Матричный метод применим к решению систем уравнений, где число уравнений равно числу неизвестных. Этот метод удобен для решения систем невысокого порядка и основан на применении свойств умножения матриц.

Пусть дана система из n уравнений с n неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = c_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = c_2, \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = c_n. \end{cases}$$

Составим матрицы:

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

свободных членов $C = \begin{Bmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{Bmatrix}$;

корней системы уравнений $X = \begin{Bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{Bmatrix}$.

Тогда систему уравнений можно записать как $A \cdot X = C$.

Массив корней системы уравнений можно найти из следующего соотношения:

$$X = A^{-1}C.$$

Таким образом, для нахождения корней системы уравнений необходимо найти обратную матрицу A^{-1}. Обратная матрица вычисляется по формуле

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

где Δ – детерминант (определитель) матрицы A, который вычисляется по формуле $\Delta = a_{11}A_{11} + \dots + a_{1j}A_{1j} + \dots + a_{1n}A_{1n}$;

A_{ij} – алгебраические дополнения элементов a_{ij}, вычисляемые по формуле $A_{ij} = (-1)^{i+j} M_{ij}$, где M_{ij} – дополнительный минор элемента a_{ij}. Дополнительный минор элемента a_{ij} является определителем союзной матрицы, получаемой из матрицы A вычеркиванием i-й строки и j-го столбца.

В результате корни системы можно вычислить по уравнению

$$X = A^{-1} \cdot C = \frac{1}{\Delta} \begin{Vmatrix} A_{11} & A_{21} & \dots & A_{n1} \\ A_{12} & A_{22} & \dots & A_{n2} \\ \dots & & & \\ A_{1n} & A_{2n} & \dots & A_{nn} \end{Vmatrix} \cdot \begin{Bmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{Bmatrix} = \begin{Bmatrix} \frac{A_{11}}{\Delta} \cdot c_1 + \frac{A_{21}}{\Delta} \cdot c_2 + \dots + \frac{A_{n1}}{\Delta} \cdot c_n \\ \frac{A_{12}}{\Delta} \cdot c_1 + \frac{A_{22}}{\Delta} \cdot c_2 + \dots + \frac{A_{n2}}{\Delta} \cdot c_n \\ \dots \\ \frac{A_{1n}}{\Delta} \cdot c_1 + \frac{A_{2n}}{\Delta} \cdot c_2 + \dots + \frac{A_{nn}}{\Delta} \cdot c_n \end{Bmatrix}.$$

Пошаговый алгоритм решения системы уравнений

Шаг 1. Сформировать матрицу A коэффициентов и вектор свободных членов C системы уравнений.

Шаг 2. Вычислить определитель матрицы A. Если он не равен нулю, следовательно, система имеет решение.

Шаг 3. Определить алгебраические дополнения $A_{11}, \dots, A_{nn}$.

Шаг 4. Построить обратную матрицу A^{-1}.

Шаг 5. Вычислить вектор X корней системы уравнений.

Пример решения системы линейных уравнений матричным методом

Постановка задачи. Решить систему линейных уравнений

$$\begin{cases} 7x_1 + 4x_2 + 3x_3 = 12; \\ 6x_1 + 4x_2 + 5x_3 = 6; \\ 4x_1 - 2x_2 + 10x_3 = -1 \end{cases}$$

матричным методом.

Пошаговый алгоритм решения

Шаг 1. Сформировать матрицу A коэффициентов системы уравнений и вектора C свободных членов: $A=\begin{Bmatrix}7 & 4 & 3\\ 6 & 4 & 5\\ 4 & -2 & 10\end{Bmatrix}$, $C=\begin{Bmatrix}12\\ 6\\ -1\end{Bmatrix}$.

Шаг 2. Вычислить определитель матрицы A:

$$\Delta = |A| = (-1)^{1+1}\cdot 7\cdot\begin{vmatrix}4 & 5\\ -2 & 10\end{vmatrix} + (-1)^{1+2}\cdot 4\cdot\begin{vmatrix}6 & 5\\ 4 & 10\end{vmatrix} + (-1)^{1+3}\cdot 3\cdot\begin{vmatrix}6 & 4\\ 4 & -2\end{vmatrix} =$$
$$= 7\cdot(4\cdot 10-(-2)\cdot 5)-4\cdot(6\cdot 10-4\cdot 5)+3\cdot(6\cdot(-2)-4\cdot 4) = 106.$$

Поскольку определитель не равен нулю, то обратная матрица существует.

Шаг 3. Вычислить алгебраические дополнения:

$$A_{11} = (-1)^{1+1}\cdot\begin{vmatrix}4 & 5\\ -2 & 10\end{vmatrix} = 4\cdot 10-(-2)\cdot 5 = 50;$$

$$A_{12} = (-1)^{1+2}\cdot\begin{vmatrix}6 & 5\\ 4 & 10\end{vmatrix} = -(6\cdot 10-4\cdot 5) = -40;$$

$$A_{13} = (-1)^{1+3}\cdot\begin{vmatrix}6 & 4\\ 4 & -2\end{vmatrix} = 6\cdot(-2)-4\cdot 4 = -28;$$

$$A_{21} = (-1)^{2+1}\cdot\begin{vmatrix}4 & 3\\ -2 & 10\end{vmatrix} = -(4\cdot 10-(-2)\cdot 3) = -46;$$

$$A_{22} = (-1)^{2+2}\cdot\begin{vmatrix}7 & 3\\ 4 & 10\end{vmatrix} = 7\cdot 10-4\cdot 3 = 58;$$

$$A_{23} = (-1)^{2+3}\cdot\begin{vmatrix}7 & 4\\ 4 & -2\end{vmatrix} = -(7\cdot(-2)-4\cdot 4) = 30;$$

$$A_{31} = (-1)^{3+1} \cdot \begin{vmatrix} 4 & 3 \\ 4 & 5 \end{vmatrix} = 4 \cdot 5 - 4 \cdot 3 = 8;$$

$$A_{32} = (-1)^{3+2} \cdot \begin{vmatrix} 7 & 3 \\ 6 & 5 \end{vmatrix} = -(7 \cdot 5 - 6 \cdot 3) = -17;$$

$$A_{33} = (-1)^{3+3} \cdot \begin{vmatrix} 7 & 4 \\ 6 & 4 \end{vmatrix} = 7 \cdot 4 - 6 \cdot 4 = 4.$$

Шаг 4. Построить обратную матрицу: $A^{-1} = \frac{1}{106} \begin{Bmatrix} 50 & -46 & 8 \\ -40 & 58 & -17 \\ -28 & 30 & 4 \end{Bmatrix}$.

Шаг 5. Вычислить вектор корней системы уравнений:

$$X = A^{-1} \cdot C = \frac{1}{106} \begin{Bmatrix} 50 & -46 & 8 \\ -40 & 58 & -17 \\ -28 & 30 & 4 \end{Bmatrix} \cdot \begin{Bmatrix} 12 \\ 6 \\ -1 \end{Bmatrix} =$$

$$= \begin{Bmatrix} \frac{50 \cdot 12}{106} + \frac{-46 \cdot 6}{106} + \frac{8 \cdot (-1)}{106} \\ \frac{-40 \cdot 12}{106} + \frac{58 \cdot 6}{106} + \frac{-17 \cdot (-1)}{106} \\ \frac{-28 \cdot 12}{106} + \frac{30 \cdot 6}{106} + \frac{4 \cdot (-1)}{106} \end{Bmatrix} = \begin{Bmatrix} 2{,}98 \\ -1{,}08 \\ -1{,}51 \end{Bmatrix}.$$

Таким образом, корни системы уравнений: $x_1 = 2{,}98$; $x_2 = -1{,}08$; $x_3 = -1{,}51$.

П 2.1.2. Метод Крамера

Суть метода Крамера заключается в вычислении корней системы уравнений с помощью формул Крамера. В этих формулах используются значения главного определителя матрицы коэффициентов системы уравнений и вспомогательных определителей, получаемых из матрицы коэффициентов заменой одного из столбцов столбцом

свободных членов. В зависимости от значений главного и вспомогательных определителей возможны следующие ситуации:

– если главный определитель системы отличен от нуля, то система имеет единственное решение;

– если главный и вспомогательные определители равны нулю, то система имеет бесконечное множество решений;

– если главный определитель равен нулю и имеется хотя бы один отличный от нуля вспомогательный определитель, то система несовместна, т. е. не имеет решения.

Пусть дана система из n уравнений с n неизвестными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = c_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = c_2, \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = c_n. \end{cases}$$

Составим матрицу коэффициентов $A = \begin{Bmatrix} a_{11}\, a_{12} \ldots a_{1n} \\ a_{21}\, a_{22} \ldots a_{2n} \\ \ldots \\ a_{n1}\, a_{n2} \ldots a_{nn} \end{Bmatrix}$;

матрицу свободных членов $C = \begin{Bmatrix} c_1 \\ c_2 \\ \ldots \\ c_n \end{Bmatrix}$;

матрицу корней системы уравнений $X = \begin{Bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{Bmatrix}$

и запишем систему уравнений в матричном виде: $A \cdot X = C$.

Если определитель матрицы системы уравнений не равен нулю, то система имеет единственное решение и это решение находится по формулам Крамера:

$$x_i = \frac{\Delta_i}{\Delta},$$

где Δ – определитель матрицы коэффициентов A;

Δ_i – определитель матрицы, получаемой из матрицы A заменой i-го столбца столбцом свободных членов C:

$$\Delta_i = \begin{vmatrix} a_{11}\, a_{12} \ldots a_{1i-1}\ c_1\ a_{1i+1} \ldots a_{1n} \\ a_{21}\, a_{22} \ldots a_{2i-1}\, c_2\ a_{2i+1} \ldots a_{2n} \\ \ldots \\ a_{n1}\, a_{n2} \ldots a_{ni-1}\ c_n\ a_{ni+1} \ldots a_{nn} \end{vmatrix}.$$

Пошаговый алгоритм решения системы уравнений

Шаг 1. Сформировать матрицу коэффициентов и вектор свободных членов системы уравнений.

Шаг 2. Вычислить определитель Δ матрицы коэффициентов системы уравнений. Если он не равен нулю, значит, система имеет решение.

Шаг 3. Вычислить первый корень системы уравнений:

– сформировать промежуточную матрицу из исходной матрицы коэффициентов A заменой первого столбца столбцом свободных членов C;

– вычислить определитель промежуточной матрицы Δ_1;

– вычислить первый корень $x_1 = \frac{\Delta_1}{\Delta}$.

Шаг 4. Вычислить остальные корни системы аналогично *шагу 3*.

Пример решения системы линейных уравнений методом Крамера

Постановка задачи. Решить систему линейных уравнений

$$\begin{cases} 7x_1 + 4x_2 + 3x_3 = 12; \\ 6x_1 + 4x_2 + 5x_3 = 6; \\ 4x_1 - 2x_2 + 10x_3 = -1 \end{cases}$$

с помощью метода Крамера.

Пошаговый алгоритм решения

Шаг 1. Сформировать матрицы коэффициентов системы и вектора свободных членов: $A = \begin{Bmatrix} 7 & 4 & 3 \\ 6 & 4 & 5 \\ 4 & -2 & 10 \end{Bmatrix}$, $C = \begin{Bmatrix} 12 \\ 6 \\ -1 \end{Bmatrix}$.

Шаг 2. Вычислить определитель матрицы A:

$$\Delta = |A| = (-1)^{1+1} \cdot 7 \cdot \begin{vmatrix} 4 & 5 \\ -2 & 10 \end{vmatrix} + (-1)^{1+2} \cdot 4 \cdot \begin{vmatrix} 6 & 5 \\ 4 & 10 \end{vmatrix} + (-1)^{1+3} \cdot 3 \cdot \begin{vmatrix} 6 & 4 \\ 4 & -2 \end{vmatrix} =$$

$$= 7 \cdot (4 \cdot 10 - (-2) \cdot 5) - 4 \cdot (6 \cdot 10 - 4 \cdot 5) + 3 \cdot (6 \cdot (-2) - 4 \cdot 4) = 106.$$

Поскольку определитель не равен нулю, то обратная матрица существует.

Шаг 3. Вычислить первый корень системы уравнений:

– сформировать промежуточную матрицу A_1 из исходной матрицы A заменой первого столбца столбцом свободных членов C:

$$A_1 = \begin{Bmatrix} 12 & 4 & 3 \\ 6 & 4 & 5 \\ -1 & -2 & 10 \end{Bmatrix};$$

– вычислить определитель матрицы A_1:

$$\Delta_1 = (-1)^{1+1} \cdot 12 \cdot \begin{vmatrix} 4 & 5 \\ -2 & 10 \end{vmatrix} + (-1)^{1+2} \cdot 4 \cdot \begin{vmatrix} 6 & 5 \\ -1 & 10 \end{vmatrix} + (-1)^{1+3} \cdot 3 \cdot \begin{vmatrix} 6 & 4 \\ -1 & -2 \end{vmatrix} =$$

$$= 12 \cdot (4 \cdot 10 - (-2) \cdot 5) - 4 \cdot (6 \cdot 10 - (-1) \cdot 5) + 3 \cdot (6 \cdot (-2) - (-1) \cdot 4) = 316;$$

– вычислить корень $x_1 = \frac{\Delta_1}{\Delta} = \frac{316}{106} = 2{,}98$.

Шаг 4. Вычислить второй корень системы уравнений:

– сформировать промежуточную матрицу A_2 из исходной матрицы A заменой второго столбца столбцом свободных членов C:

$$A_2 = \begin{Bmatrix} 7 & 12 & 3 \\ 6 & 6 & 5 \\ 4 & -1 & 10 \end{Bmatrix};$$

– вычислить определитель матрицы A_2:

$$\Delta_2 = (-1)^{1+1} \cdot 7 \cdot \begin{vmatrix} 6 & 5 \\ -1 & 10 \end{vmatrix} + (-1)^{1+2} \cdot 12 \cdot \begin{vmatrix} 6 & 5 \\ 4 & 10 \end{vmatrix} + (-1)^{1+3} \cdot 3 \cdot \begin{vmatrix} 6 & 6 \\ 4 & -1 \end{vmatrix} =$$

$$= 7 \cdot (6 \cdot 10 - (-1) \cdot 5) - 12 \cdot (6 \cdot 10 - 4 \cdot 5) + 3 \cdot (6 \cdot (-1) - 4 \cdot 6) = -115;$$

– вычислить второй корень:

$$x_2 = \frac{\Delta_2}{\Delta} = \frac{-115}{106} = -1{,}08.$$

Шаг 5. Вычислить третий корень системы уравнений:

– сформировать промежуточную матрицу A_3 из исходной матрицы A заменой третьего столбца столбцом свободных элементов C:

$$A_3 = \begin{Bmatrix} 7 & 4 & 12 \\ 6 & 4 & 6 \\ 4 & -2 & -1 \end{Bmatrix};$$

– вычислить определитель матрицы A_3:

$$\Delta_3 = (-1)^{1+1} \cdot 7 \cdot \begin{vmatrix} 4 & 6 \\ -2 & -1 \end{vmatrix} + (-1)^{1+2} \cdot 4 \cdot \begin{vmatrix} 6 & 6 \\ 4 & -1 \end{vmatrix} + (-1)^{1+3} \cdot 12 \cdot \begin{vmatrix} 6 & 4 \\ 4 & -2 \end{vmatrix} =$$

$$= 7 \cdot (4 \cdot (-1) - (-2) \cdot 6) - 4 \cdot (6 \cdot (-1) - 4 \cdot 6) + 12 \cdot (6 \cdot (-2) - 4 \cdot 4) = -160;$$

– вычислить третий корень:

$$x_3 = \frac{\Delta_3}{\Delta} = \frac{-160}{106} = -1{,}51.$$

Таким образом, корни системы уравнений: $x_1 = 2{,}98$; $x_2 = -1{,}08$; $x_3 = -1{,}51$.

П 2.1.3. Метод Гаусса

При численном решении систем линейных уравнений одним из распространенных является метод Гаусса, сущность которого сводится к поэтапному исключению неизвестных из уравнений.

Решение системы уравнений методом Гаусса возможно только в случае, если матрица коэффициентов системы уравнений невырожденная, т. е. её определитель (детерминант) не равен нулю.

Пусть необходимо решить систему линейных уравнений

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = c_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = c_2, \\ \ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = c_n. \end{cases} \qquad (\text{П } 2.1)$$

Пошаговый алгоритм решения

Процесс решения делится на два этапа. На первом этапе (прямой ход) последовательным исключением неизвестных составляется преобразованная эквивалентная система уравнений с треугольной матрицей коэффициентов, в которой все элементы под главной диагональю равны нулю, т. е. добиваются того, что в первом уравнении остаются все переменные, а в каждом последующем убирается по одной переменной до тех пор, пока в последнем уравнении не останется только одна переменная.

На втором этапе (обратный ход) решается полученная система уравнений. С помощью подстановок проводится последовательное вычисление неизвестных переменных. Решение начинается с по-

следнего уравнения и заканчивается решением первого уравнения системы.

Шаг 1. Среди элементов a_{ij} ($i, j = 1, ..., n$) матрицы A необходимо выбрать главный элемент a_{pq}, являющийся наибольшим по модулю. Соответствующая строка матрицы A с номером p называется главной строкой. Главную строку необходимо поместить на место первого уравнения, для чего следует поменять местами первую строку со строкой p. Главный элемент должен находиться на месте первого коэффициента уравнения, для этого надо поменять местами первый столбец коэффициентов со столбцом q. В результате главный элемент окажется на месте элемента a_{11}.

При написании программы необходимо запоминать информацию о старой нумерации коэффициентов и вводить новую нумерацию.

Шаг 2. Необходимо, чтобы коэффициент первого уравнения a_{11} был равен 1. Для этого следует разделить коэффициенты первого уравнения системы на значение $a_{11} = a_{pq}$, в результате первое уравнение примет вид $x_1 + c_{12}x_2 + .. + c_{1n} = d_1$, где $c_{1j} = a_{1j} / a_{11}$; $d_1 = b_1 / a_{11}$.

Шаг 3. Теперь следует исключить неизвестную переменную x_1 из каждого уравнения исходной системы, начиная со второго. Для этого выполняются следующие действия:

1) умножая коэффициенты первого уравнения на первый коэффициент второго уравнения (a_{12}), получают преобразованное первое уравнение, которое и вычитают из второго уравнения (первое уравнение остается неизменным);

2) умножая коэффициенты первого уравнения на первый коэффициент третьего уравнения (a_{13}), получают преобразованное первое уравнение, которое вычитают из третьего уравнения (первое уравнение остается неизменным);

…

n–1) вычетают из n-го уравнения произведение первого уравнения и первого коэффициента n-ного уравнения (a_{1n}). В результате получают следующую систему уравнений:

$$\begin{cases} x_1 + c_{12}x_2 + c_{13}x_3 + \ldots + c_{1n}x_n = d_1; \\ \qquad x_2 + c_{23}x_3 + \ldots + c_{2n}x_n = d_2; \\ \qquad x_2 + c_{23}x_3 + \ldots + c_{3n}x_n = d_3; \\ \qquad\qquad \ldots \\ \qquad x_2 + c_{23}x_3 + \ldots + c_{nn}x_n = d_n. \end{cases}$$

Шаг 4. Последовательно повторяют шаги 2–3 для исключения переменных от x_2 до x_{n-1}. В результате получают систему уравнений с треугольной матрицей, эквивалентной исходной (П 2.1):

$$\begin{cases} x_1 + c_{12}x_2 + c_{13}x_3 + \ldots + c_{1n}x_n = d_1, \\ \qquad x_2 + c_{23}x_3 + \ldots + c_{2n}x_n = d_2, \\ \qquad\qquad x_3 + \ldots + c_{3n}x_n = d_3, \\ \qquad\qquad\quad \ldots \\ \qquad\qquad\qquad c_{nn}x_n = d_n. \end{cases} \qquad (\text{П } 2.2)$$

Шаг 5. Обратный ход. Неизвестные переменные $x_1, x_2, .., x_n$ определяют в обратном порядке по формулам, полученным из системы уравнений (П 2.2):

$$x_n = d_n / c_{nn};$$

$$x_{n-1} = d_{n-1} - c_{n-1,n}x_n;$$

$$\ldots$$

$$x_1 = d_1 - c_{1,2}x_2 - .. - c_{1,n}x_n.$$

Пример решения системы линейных уравнений методом Гаусса

Постановка задачи. Решить систему уравнений

$$\begin{cases} 7x_1 + 4x_2 + 3x_3 = 12; \\ 6x_1 + 4x_2 + 5x_3 = 6; \\ 4x_1 - 2x_2 + 10x_3 = -1 \end{cases} \qquad (\text{П } 2.3)$$

методом Гаусса.

Пошаговый алгоритм решения

Прямой ход. *Шаг 1*. Найти максимальный коэффициент системы уравнений. В системе уравнений (П 2.3) $a_{pq} = a_{33} = 10$. Необходимо переместить максимальный коэффициент на место коэффициента a_{11}. Для этого необходимо произвести следующие действия:

1) поменять местами первую и третью строки:

$$\begin{cases} 4x_1 - 2x_2 + 10x_3 = -1; \\ 6x_1 + 4x_2 + 5x_3 = 6; \\ 7x_1 + 4x_2 + 3x_3 = 12; \end{cases}$$

2) поменять местами первый и третий столбцы:

$$\begin{cases} 10x_3 - 2x_2 + 4x_1 = -1; \\ 5x_3 + 4x_2 + 6x_1 = 6; \\ 3x_3 + 4x_2 + 7x_1 = 12. \end{cases}$$

Шаг 2. Разделить коэффициенты первого уравнения (первая строка) на первый коэффициент строки (число 10):

$$\begin{cases} x_3 - 0{,}2x_2 + 0{,}4x_1 = -0{,}1; \\ 5x_3 + 4x_2 + 6x_1 = 6; \\ 3x_3 + 4x_2 + 7x_1 = 12. \end{cases}$$

Шаг 3. Исключить переменную x_3 из второго и третьего уравнений. Для этого произвести следующие действия.

1) Из второго уравнения вычесть произведение первого уравнения и первого коэффициента второго уравнения (число 5). Вместо второго уравнения системы будет получено уравнение $x_3(5-5) + x_2(4+0{,}2\cdot 5) + x_1(6-0{,}4\cdot 5) = 6+0{,}1\cdot 5.$

После приведения подобных членов второе уравнение примет вид $5x_2 + 4x_1 = 6{,}5$.

2) Из третьего уравнения вычесть произведение первого уравнения и первого коэффициента третьего уравнения (число 3). Вместо

третьего уравнения системы будет получено уравнение $x_3(3-3)+x_2(4+0{,}2\cdot 3)+x_1(7-0{,}4\cdot 3)=12+0{,}1\cdot 3$.

После приведения подобных членов третье уравнение примет вид $4{,}6x_2+5{,}8x_1=12{,}3$.

В результате будет получена система уравнений:

$$\begin{cases} x_3-0{,}2x_2+0{,}4x_1=-0{,}1; \\ 5x_2+4x_1=6{,}5; \\ 4{,}6x_2+5{,}8x_1=12{,}3. \end{cases} \qquad (\text{П } 2.4)$$

Шаг 4. На этом шаге будем работать со вторым и третьим уравнениями. Найти максимальный коэффициент системы уравнений (П 2.4): $a_{33} = 5{,}8$. Поместить коэффициент a_{33} на место коэффициента a_{21}. Произвести действия, аналогичные *шагу 1*:

1) поменять местами вторую и третью строки:

$$\begin{cases} x_3-0{,}2x_2+0{,}4x_1=-0{,}1; \\ 4{,}6x_2+5{,}8x_1=12{,}3; \\ 5x_2+4x_1=6{,}5; \end{cases}$$

2) поменять местами второй и третий столбцы:

$$\begin{cases} x_3+0{,}4x_1-0{,}2x_2=-0{,}1; \\ 5{,}8x_1+4{,}6x_2=12{,}3; \\ 4x_1+5x_2=6{,}5. \end{cases}$$

Шаг 5. Исключить переменную x_1 из третьего уравнения. Для этого произвести следующие действия.

1) Второе уравнение разделить на значение первого коэффициента второго уравнения a_{21} (число 5,8):

$$\begin{cases} x_3+0{,}4x_1-0{,}2x_2=-0{,}1; \\ x_1+0{,}79x_2=2{,}12; \\ 4x_1+5x_2=6{,}5. \end{cases}$$

2) Из третьего уравнения вычесть произведение второго уравнения и первого коэффициента третьего уравнения (число 4). Вместо третьего уравнения системы будет получено $(4-4)x_1 + (5-0{,}79\cdot 4)x_2 = 6{,}5-2{,}12\cdot 4$.

После упрощения третье уравнение примет вид $1{,}84x_2 = -1{,}98$.

В результате будет получена система уравнений

$$\begin{cases} x_3 + 0{,}4x_1 - 0{,}2x_2 = -0{,}1; \\ x_1 + 0{,}79x_2 = 2{,}12; \\ 1{,}84x_2 = -1{,}98. \end{cases}$$

Таким образом, система приведена к треугольному виду, в котором все коэффициенты под главной диагональю равны нулю.

Обратный ход. Выполнить вычисление переменных последовательно, начиная с третьего и заканчивая первым уравнением.

Шаг 6. Получить x_2 из третьего уравнения:

$$\begin{cases} x_3 + 0{,}4x_1 - 0{,}2x_2 = -0{,}1; \\ x_1 + 0{,}79x_2 = 2{,}12; \\ x_2 = \dfrac{-1{,}98}{1{,}84} = -1{,}08. \end{cases}$$

Шаг 7. Получить x_1 из второго уравнения:

$$\begin{cases} x_3 + 0{,}4x_1 - 0{,}2x_2 = -0{,}1; \\ x_1 = 2{,}12 - 0{,}79\cdot(-1{,}08) = 2{,}98; \\ x_2 = -1{,}08. \end{cases}$$

Шаг 8. Получить x_3 из первого уравнения:

$$\begin{cases} x_3 = -0{,}1 - 0{,}4\cdot 2{,}98 + 0{,}2\cdot(-1{,}08) = -1{,}51; \\ x_1 = 2{,}98; \\ x_2 = -1{,}08. \end{cases}$$

Таким образом, корни системы уравнений: $x_1 = 2{,}98$; $x_2 = -1{,}08$; $x_3 = -1{,}51$.

П 2.2. Основные теоретические сведения о численных методах решения нелинейных уравнений

Общая постановка задачи. Во многих практических инженерных задачах возникает необходимость решения уравнения вида

$$f(x, p_1, p_2, \ldots, p_n) = 0, \qquad (\text{П } 2.5)$$

где f – заданная функция;

x – неизвестная величина;

$p_1, p_2, \ldots, p_n$ – параметры задачи.

Например, уравнение $e^{4x} + x^5 = \ln(\cos(x))$ не имеет решения в аналитическом виде, так как решение нельзя выразить в виде алгебраических функций.

Только для простейших нелинейных уравнений можно найти решение в аналитическом виде, т. е. записать формулу, выражающую искомую величину x в явном виде через параметры p_k. В большинстве же случаев уравнения вида (П 2.5) приходится решать численными методами. В результате численного решения уравнения (П 2.5) получают таблицы зависимостей искомой величины x от параметров p_k. Чаще всего фиксируют все параметры, за исключением одного, который изменяют в интересующем интервале с выбранным шагом и получают одномерные таблицы и графики на плоскости.

Численное решение уравнения (П 2.5) обычно проводят в два этапа. На первом этапе необходимо отделить корни уравнения, т. е. найти такие интервалы изменения переменной x, где расположен только один корень. Для этого в определенном интервале изменения переменной $x \in [x_0, x_n]$ при фиксированных параметрах p_k необходимо вычислить ряд значений $f(x)$ и результаты поместить в таблицу. На втором этапе находят корни с заданной точностью. Для уточнения корней могут использоваться методы дихотомии, методы Ньютона и др.

В основе этих методов лежат рекуррентные последовательности. Рекуррентная последовательность – это последовательность, члены

которой связаны рекуррентной формулой. Рекуррентная формула – это формула вида

$$a_n = f(n, a_{n-1}, a_{n-2}, ..., a_{n-p}),\ n \geq p+1,$$

выражающая каждый член последовательности a_n ($n \in N$) через p предыдущих членов.

Общая постановка задачи. Функция $f(x)$ задана на отрезке $[a, b]$. Необходимо с заданной погрешностью ε найти корень уравнения

$$f(x) = 0. \qquad (\text{П } 2.6)$$

П 2.2.1. Метод дихотомии

Сущность метода дихотомии состоит в поэтапном делении отрезка $[a, b]$, на котором находится корень уравнения (П 2.6), пополам.

На первом этапе табулируют функцию на отрезке $[x_0, x_n]$ с заданным шагом. В результате получают таблицу значений

x	x_0	x_1	x_2	...	x_n
$f(\ )$	$f(x_0)$	$f(x_1)$	$f(x_2)$	...	$f(x_n)$

Из таблицы выделяют интервалы $[x_i, x_{i+1}]$, на которых функция изменяет свой знак ($f(x_i) > 0$ и $f(x_{i+1}) < 0$), что проиллюстрировано на рис. П 2.1.

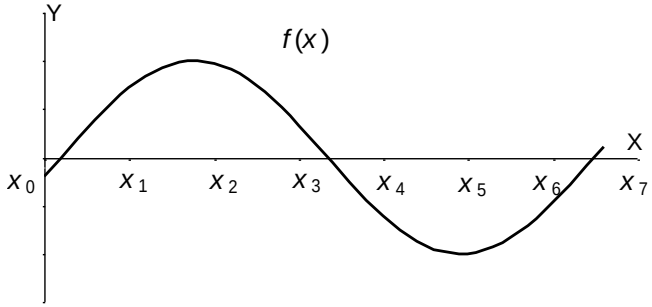
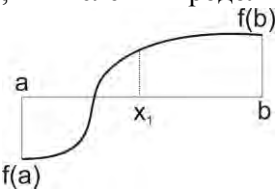


Рис. П 2.1

Из рисунка видно, что на отрезках $[x_0, x_1]$, $[x_3, x_4]$, $[x_6, x_7]$ функция меняет свой знак, следовательно, на интервале $[x_0, x_7]$ она имеет три корня. На втором этапе на выделенных интервалах осуществляется поиск корней с заданной точностью ε.

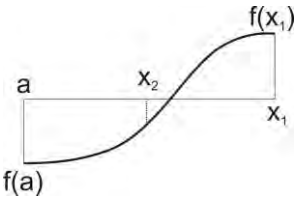
Пошаговый алгоритм решения

Шаг 1. Определяется первое приближение корня $x_1 = (a + b)/2$, которое затем проверяется на точность (условие окончания вычислений). Если $(b - a)/2 > \varepsilon$, вычисления продолжаются.



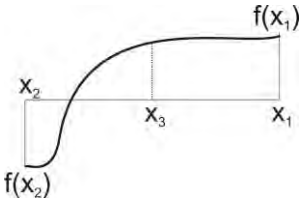
Шаг 2. Из двух отрезков $[a, x_1]$ и $[x_1, b]$ выбирается тот, на котором функция $f(x)$ меняет знак. В соответствии с иллюстрацией выбирается отрезок $[a, x_1]$, на котором выполняется условие $f(a) \cdot f(x_1) < 0$.

Шаг 3. Вычисляется второе приближение корня $x_2 = (a + x_1)/2$ и проверяется условие окончания вычислений. Если $(x_1 - a)/2 > \varepsilon$, вычисления продолжаются.



Шаг 4. Из двух отрезков $[a, x_2]$ и $[x_2, x_1]$ выбирается тот, на котором функция $f(x)$ меняет знак. В соответствии с иллюстрацией выбирается отрезок $[x_2, x_1]$, на котором выполняется условие $f(x_2) \cdot f(x_1) < 0$.

Шаг 5. Вычисляется третье приближение корня $x_3 = (x_1 + x_2)/2$ и проверяется условие окончания вычислений.



…

Шаг m. Вычисляется *n*-е приближение корня $x_n = (x_{n-1} + x_{n-2})/2$ и проверяется условие окончания вычислений. Если $(x_{n-1} - x_{n-2})/2 \le \varepsilon$, вычисления закончены и корнем нелинейного уравнения является значение x_n.

П 2.2.2. Методы Ньютона (метод хорд и метод касательных)

Методы Ньютона (метод хорд и метод касательных) являются более эффективными методами решения нелинейных уравнений по сравнению с методом дихотомии. Сходимость методов достигается при удовлетворении следующих условий:

1) функция $f(x)$ должна быть дважды дифференцируема на отрезке $[a, b]$;

2) обе производные функции $f(x)$ (и первая и вторая) не меняют знак на отрезке $[a, b]$.

Метод хорд. При использовании метода хорд проводится хорда, соединяющая точки графика функции $f(x)$ на границах отрезка, и за новое приближение берется точка, в которой хорда пересекается осью X. Новое приближение корня x_i вычисляется по формуле $x_i = \dfrac{b \cdot f(a) - a \cdot f(b)}{f(a) - f(b)}$. Вычисления продолжаются до тех пор, пока разница между приближениями не станет меньше или равна погрешности, а именно $|x_{i+1} - x_i| \le \varepsilon$.

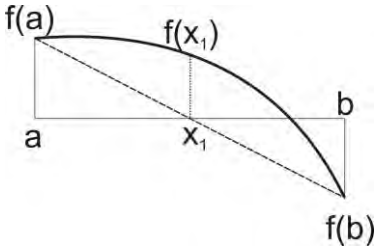
В иллюстрациях пошагового алгоритма пунктирной линией обозначена проведенная хорда.

Пошаговый алгоритм решения

Шаг 1. Определяется первое приближение корня

$$x_1 = \frac{b \cdot f(a) - a \cdot f(b)}{f(a) - f(b)}.$$

Проверяется условие окончания вычислений. Если $|b - a| > \varepsilon$, вычисления продолжаются.

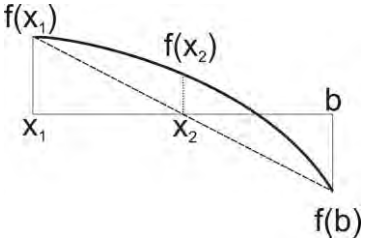


Шаг 2. Из двух отрезков $[a, x_1]$ и $[x_1, b]$ выбирается тот, на котором функция $f(x)$ меняет знак. В соответствии с иллюстрацией выбирается отрезок $[x_1, b]$, на котором выполняется условие $f(b) \cdot f(x_1) < 0$.

Шаг 3. Вычисляется второе приближение корня

$$x_2 = \frac{b \cdot f(x_1) - x_1 \cdot f(b)}{f(x_1) - f(b)}$$

и проверяется условие окончания вычислений. Если $|x_1 - b| > \varepsilon$, вычисления продолжаются.



Шаг 4. Из двух отрезков $[x_1, x_2]$ и $[x_2, b]$ выбирается тот, на котором функция $f(x)$ меняет знак. В соответствии с иллюстрацией выбирается отрезок $[x_2, b]$, на котором выполняется условие $f(b)\ f(x_2) < 0$.

Шаг 5. Вычисляется третье приближение корня

$$x_3 = \frac{b \cdot f(x_2) - x_2 \cdot f(b)}{f(x_2) - f(b)}$$

и проверяется условие окончания вычислений. Если $|x_2 - b| > \varepsilon$, вычисления продолжаются.

…

Шаг m. Вычисляется n-е приближение корня на отрезке $[x_{n-2}, x_{n-1}]$

$$x_n = \frac{x_{n-2} \cdot f(x_{n-1}) - x_{n-1} \cdot f(x_{n-2})}{f(x_{n-1}) - f(x_{n-2})}$$

и проверяется условие окончания вычислений. Если $|x_{n-1} - x_{n-2})| \le \varepsilon$, вычисления закончены и корнем нелинейного уравнения является значение x_n.

Метод касательных. При использовании метода касательных проводится касательная к графику функции $f(x)$ и за новое приближение принимается точка, в которой касательная пересекает ось X.

Новое приближение корня вычисляется по формуле $x_{i+1} = c - \frac{f(c)}{f'(c)}$. В качестве точки c берется точка начала или конца рассматриваемого отрезка, в которой функция $f(c)$ и вторая производная $f''(c)$ имеют одинаковые знаки. Для проверки этого условия используется выражение $f(c) \cdot f''(c)$. Если оно больше нуля, то эти величины имеют одинаковые знаки, если меньше – разные. Вычисления продолжаются до тех пор, пока разница между приближениями не станет меньше или равна погрешности, а именно $|x_{i+1} - x_i| \le \varepsilon$.

В иллюстрациях пошагового алгоритма пунктирной линией обозначена проведенная касательная.

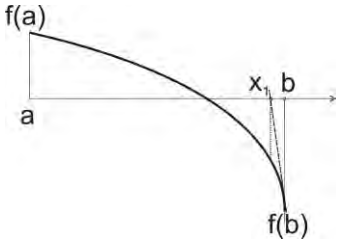
Пошаговый алгоритм решения

Шаг 1. Определяется точка для проведения касательной – это точка начала (a) или конца отрезка (b). Поскольку в точке b функ-

ция $f(b)$ и вторая производная $f''(b)$ имеют одинаковые знаки (выполняется условие $f(b) \cdot f''(b) > 0$), то искомая точка b.

Шаг 2. Вычисляется первое приближение к корню $x_1 = b - \dfrac{f(b)}{f'(b)}$.

Проверяется условие окончания вычислений. Если $|b - a| > \varepsilon$, вычисления продолжаются.



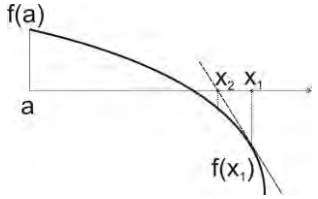
Шаг 3. Из двух отрезков $[a, x_1]$ и $[x_1, b]$ выбирается тот, на котором функция $f(x)$ меняет знак. В соответствии с иллюстрацией выбирается отрезок $[a, x_1]$, на котором выполняется условие $f(a) \cdot f(x_1) < 0$.

Шаг 4. Определяется точка для проведения касательной – это точка начала (a) или конца отрезка (x_1). Поскольку в точке x_1 функция $f(x_1)$ и вторая производная $f''(x_1)$ имеют одинаковые знаки (выполняется условие $f(x_1) \cdot f''(x_1) > 0$), искомая точка x_1.

Шаг 5. Вычисляется второе приближение к корню

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}.$$

Проверяется условие окончания вычислений. Если $|x_2 - x_1| > \varepsilon$, вычисления продолжаются.



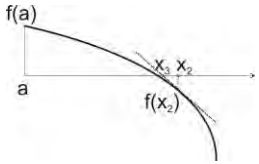
Шаг 7. Из двух отрезков $[a, x_2]$ и $[x_2, x_1]$ выбирается тот, на котором функция $f(x)$ меняет знак. В соответствии с иллюстрацией выбирается отрезок $[a, x_2]$, на котором выполняется условие $f(a)\cdot f(x_2) < 0$.

Шаг 8. Определяется точка для проведения касательной – это точка начала (a) или конца отрезка (x_2). Поскольку в точке x_2 функция $f(x_2)$ и вторая производная $f''(x_2)$ имеют одинаковые знаки (выполняется условие $f(x_2)\cdot f''(x_2) > 0$), искомая точка – x_2.

Шаг 9. Вычисляется третье приближение к корню

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}.$$

Проверяется условие окончания вычислений. Поскольку $|x_3 - x_2| > \varepsilon$, вычисления продолжаются.



…

Шаг m. Рассматривается отрезок $[x_{n-2}, x_{n-1}]$. Определяется точка для проведения касательной – это точка начала (x_{n-2}) или конца отрезка (x_{n-1}). Поскольку в точке x_{n-2} функция $f(x_{n-2})$ и вторая производная $f''(x_{n-2})$ имеют одинаковые знаки (выполняется условие $f(x_{n-2})\cdot f''(x_{n-2}) > 0$), искомая точка x_{n-2}.

Шаг $m+1$. Вычисляется n-е приближение корня на отрезке $[x_{n-2},\ x_{n-1}]$ $x_n = x_{n-2} - \dfrac{f(x_{n-2})}{f'(x_{n-2})}$ и проверяется условие окончания вычислений. Если $|x_{n-1} - x_{n-2}| \le \varepsilon$, вычисления закончены и корнем нелинейного уравнения является значение x_n.

П 2.2.3. Метод простых итераций

Пусть задан отрезок $[a, b]$ для поиска корня функции $f(x)$. Представим уравнение (П 2.6) в виде

$$x = \varphi(x). \qquad (\text{П } 2.7)$$

Если для всех точек $p \in [a,b]$ выполняются два условия:

1) функция $\varphi(x) \in [a,b]$;

2) производная $|\varphi'(x)| < 1$,

то существует единственное решение уравнения (П 2.7) при любом начальном значении корня $x_0 \in [a,b]$.

Для поиска корня необходимо использовать итерационную формулу: каждое следующее приближения корня x_{i+1} вычисляется по формуле $x_{i+1} = \varphi(x_i)$. Вычисления продолжаются до тех пор, пока разница между приближениями не станет меньше погрешности, а именно $|x_{i+1} - x_i| \le \varepsilon$.

П 2.3. **Основные теоретические сведения о численных методах аппроксимации**

Методы аппроксимации функции используются для определения аналитической формулы многочлена, который наилучшим образом описывает исходную функцию, заданную таблично.

Общая постановка задачи. Пусть функция $y_i = f(x_i)$, где $i = 1, n$, задана таблично. Определить аппроксимационный многочлен m-степени, наилучшим образом описывающий табличную функцию. В качестве критерия оптимизации использовать функционал, построенный по методу наименьших квадратов и обеспечивающий минимальный разброс точек табличной и моделируемой функций многочлена. Вид функционала следующий:

$$S = \sum_{i=1}^{n} [P_m(x_i) - y_i]^2,$$

где y_i – функция, заданная таблично;

$P_m(x_i)$ – многочлен вида $P_m = \sum_{i=0}^{m} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m$;

n – число точек табличной функции;

m – степень аппроксимирующего многочлена.

Для определения оптимального многочлена, т. е. многочлена, удовлетворяющего условию $S \to \min$, определим частные производные функционала

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 2\sum\limits_{i=1}^{n} (a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i) = 0, \\ \dfrac{\partial S}{\partial a_1} = 2\sum\limits_{i=1}^{n} (a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i) \cdot x_i = 0, \\ \ldots \\ \dfrac{\partial S}{\partial a_m} = 2\sum\limits_{i=1}^{n} (a_0 + a_1 x_i + \ldots + a_m x_i^m - y_i) \cdot x_i^m = 0. \end{cases} \qquad (\text{П } 2.8)$$

После преобразований со знаками сумм система (П 2.8) примет вид

$$\begin{cases} \sum\limits_{i=1}^{n} (a_0 + a_1 x_i + \ldots + a_m x_i^m) = \sum\limits_{i=1}^{n} y_i, \\ \sum\limits_{i=1}^{n} (a_0 x_i + a_1 x_i^2 + \ldots + a_m x_i^{m+1}) = \sum\limits_{i=1}^{n} x_i y_i, \\ \ldots \ldots \ldots \ldots \ldots \\ \sum\limits_{i=1}^{n} (a_0 x_i^m + a_1 x_i^{m+1} + \ldots + a_m x_i^{2m}) = \sum\limits_{i=1}^{n} x_i^m y_i. \end{cases} \qquad (\text{П } 2.9)$$

Введем обозначения $t_0 = n$; $t_1 = \sum\limits_{i=1}^{n} x_i$; …, $t_m = \sum\limits_{i=1}^{n} x_i^m$; $c_0 = \sum\limits_{i=1}^{n} y_i$; $c_1 = \sum\limits_{i=1}^{n} x_i y_i$; …, $c_m = \sum\limits_{i=1}^{n} x_i^m y_i$.

В новых переменных система (П 2.9) имеет вид

$$\begin{cases} t_0 a_0 + t_1 a_1 + \ldots + t_m a_m = c_0, \\ t_1 a_0 + t_2 a_1 + \ldots + t_{m+1} a_m = c_1, \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \\ t_m a_0 + t_{m+1} a_1 + \ldots + t_{2m} a_m = c_m. \end{cases} \qquad (\text{П } 2.10)$$

Решая систему линейных уравнений (П 2.10), можно определить коэффициенты многочлена $a_0,\ a_1, a_2, ..., a_m$.

Схема алгоритма аппроксимации представлена на рис. П 2.2.

Рис. П 2.2

П 2.4. Основные теоретические сведения о численных методах дифференцирования табулированных функций

При решении математических уравнений численными методами возникает необходимость в вычислении производной. В основе численного дифференцирования лежит определение производной:

$$y' = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}.$$

При вычислении производной в j-й точке используются различные приближения:

а) центрально-разностное

$$y'\big|_{x=x_j} = \frac{y_{j+1} - y_{j-1}}{x_{j+1} - x_{j-1}}; \qquad (\text{П } 2.11)$$

б) левое разностное

$$y'\big|_{x=x_j} = \frac{y_j - y_{j-1}}{x_j - x_{j-1}}; \qquad (\text{П } 2.12)$$

в) правое разностное

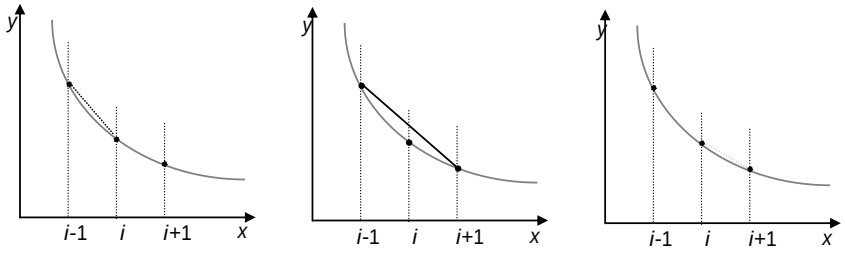
$$y'\big|_{x=x_j} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j}. \qquad (\text{П } 2.13)$$

Иллюстрация геометрического смысла численного дифференцирования представлена на рис. П 2.3, где изображены элементы, используемые при построении левых (*а*), центральных (*б*) и правых (*в*) разностей.

Иллюстрация геометрического смысла численного дифференцирования представлена на рис. П 2.3, где изображены элементы, используемые при построении левых (*а*), центральных (*б*) и правых (*в*) разностей.

Для построения второй производной чаще всего используют центрально-разностное приближение

$$y''\big|_{x=x_j} = \frac{y_{j+1} - 2y_j + y_{j-1}}{(\Delta x)^2}. \qquad (\text{П } 2.14)$$



а *а*) *б*) *в*) *в*

Рис. П 2.3

Левые и правые разности функции соответственно строятся по формулам

Для построения второй производной чаще всего используют центрально-разностное приближение

$$y''\big|_{x=x_j} = \frac{y_j - 2y_{j-1} + y_{j-2}}{(\Delta x)^2};$$

$$y''\big|_{x=x_j} = \frac{y_{j+2} - 2y_{j+1} + y_j}{(\Delta x)^2}.$$

Замена производной конечными разностями приводит к некоторым погрешностям, так как происходит сглаживание нелинейных участков функции. Чем меньше шаг изменения аргумента, тем больше точность аппроксимации функции конечными разностями. Считается, что формула центральных разностей дает более точное численное значение производной, чем формула, соответствующая левым и правым разностям.

При вычислении производных второго, третьего, четвертого и более высоких порядков наиболее удобно пользоваться конечно-разностной аппроксимацией первой производной. При этом вычисление второй производной основывается на конечных разностях первой производной. Для вычисления третьей производной необходимо использовать конечно-разностную аппроксимацию, составленную из вторых производных. Общая формула при этом имеет вид

$$f^{n}(\ ) = \frac{f^{n-1}(\ ) - f^{n-1}(\ )}{x_{j+1} - x_{j-1}}, \qquad (\text{П } 2.15)$$

где n – порядковый номер производной.

П 2.5. **Основные теоретические сведения о численных методах интегрирования табулированных функций**

Методы численного интегрирования функций основываются на специальных формулах суммирования подынтегральных функций. Геометрический смысл вычисления интеграла основан на формуле

$$\int_{x_0}^{x_n} f(x)\,dx = \sum_{i=1}^{n} A_i f(x_i),$$

где $f(x)$ – подынтегральная функция;

$A_i f(x_i)$ – квадратурная сумма;

A_i – коэффициенты квадратурной суммы;

x_i – узлы квадратурной функции;

x_0, x_n – пределы интегрирования функции;

i – индекс, связанный с разбиением функции пошаговым методом;

n – число разбиений функции.

На рис. П 2.4 представлена иллюстрация метода вычисления определенного интеграла по формуле прямоугольников ($h = x_{i+1} - x_i$ – шаг интегрирования: $x_1 = x_0 + h$, …, $x_{i+1} = x_i + h$, …, $x_n = x_{n-1} + h$).

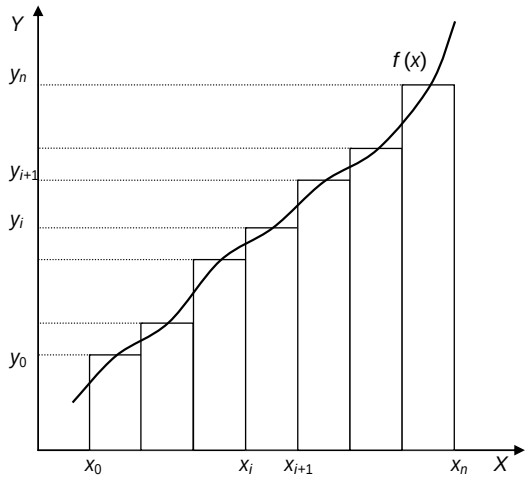


Рис. П 2.4

Для вычисления интегралов используются:

а) формула прямоугольников

$$\int_{x_0}^{x_n} f(x)dx = h\left[f\left(x_0 + \frac{h}{2}\right) + f\left(x_0 + \frac{3}{2}h\right) + \ldots + f\left(x_0 + \frac{2k+1}{2}h\right)\right], \quad (\text{П } 2.16)$$

где $h = \dfrac{x_n - x_0}{k}$ (k – число разбиений);

h – шаг интегрирования;

x_0, x_n – пределы интегрирования;

б) формула трапеций

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{2}\left[f(x_0) + 2f(x_0+h) + \ldots + 2f(x_0+h(n-1)) + f(x_n)\right] \quad (\text{П } 2.17)$$

(ошибка вычисления интегралов (П 2.16), (П 2.17) определяется как $|\varepsilon| \le \frac{h^2}{12}(x_n - x_0)\max f''(\xi)$, где $x_0 < \xi < x_n$);

в) формула Симпсона

$$\int_{x_0}^{x_n} f(x)dx = \frac{h}{3}\left[f(x_0) + 4f(x_0+h) + 2f(x_0+2h) + \ldots + 2f(x_0+(n-2)h) + 4f(x_0+(n-1)h) + f(x_0+nh)\right] \quad (\text{П } 2.18)$$

(ошибка вычисления интеграла (П 2.18) определяется по формуле

$$|\varepsilon| \approx \frac{h^4}{180}(x_n - x_0)\max f^{IV}(\xi)$$

где $x_0 < \xi < x_n$,

$f^{IV}(\xi)$ – четвертая производная функции).

Приведем примеры вычисления интеграла $\int_4^9 \cos(3x)dx$ различными методами, приняв шаг интегрирования равным 0,01; число разбиений интервала интегрирования – равным 500:

а) по формуле прямоугольников:

$$\int_4^9 \cos(3x)dx = 0{,}01\left[\cos\left(3\left(4+\frac{0{,}01}{2}\right)\right) + \cos\left(3\left(4+\frac{3}{2}\cdot 0{,}01\right)\right) + \ldots + \cos\left(3\left(4+\frac{2\cdot 500+1}{2}\cdot 0{,}01\right)\right)\right],$$

ошибка вычисления интеграла $|\varepsilon| \approx 0{,}125\cdot 10^{-3}$;

б) по формуле трапеций:

$$\int_4^9 \cos(3x)dx = \frac{0{,}01}{2}\left[\cos(3\cdot 4) + 2\cos(3(4+0{,}01)) + 2\cos(3(4+2\cdot 0{,}01)) + \ldots + \right.$$
$$\left. + 2\cos(3(4+499\cdot 0{,}01)) + \cos(3(4+500\cdot 0{,}01))\right]$$

ошибка вычисления интеграла $|\varepsilon| \approx 0{,}125\cdot 10^{-3}$;

в) по формуле Симпсона:

$$\int_4^9 \cos(3x)dx = \frac{0{,}01}{3}\left[\cos(3\cdot 4) + 4\cos(3(4+0{,}01)) + 2\cos(3(4+2\cdot 0{,}01)) + \ldots + \right.$$
$$\left. + 2\cos(3(4+498\cdot 0{,}01)) + 4\cos(3(4+499\cdot 0{,}01)) + \cos(3(4+500\cdot 0{,}01))\right]$$

ошибка вычисления интеграла $|\varepsilon| \approx 0{,}225\cdot 10^{7}$.

П 2.6. **Основные теоретические сведения о численных методах решения обыкновенных дифференциальных уравнений**

Дифференциальным уравнением первого порядка называется уравнение вида

$$y' = f(x, y). \qquad (\text{П } 2.19)$$

Решить дифференциальное уравнение – означает определить функцию $y = F(x)$, которая при подстановке в уравнение (П 2.19) приводит к тождеству. Поиск решения $y = F(x)$ при реализации аналитических методов сводится к интегрированию уравнения (П 2.19), что не всегда приводит к успеху из-за сложности функции $f(x, y)$. В этом случае для решения дифференциальных уравнений $y' = f(x, y)$ используют численные методы.

Для решения уравнения (П 2.19) вводят начальное условие Коши $y(x_0) = y_0$.

Таким образом, общей постановкой задачи для решения дифференциальных уравнений первого порядка является

$$y' = f(x, y),$$
$$y(x_0) = y_0. \qquad (П\ 2.20)$$

Для решения дифференциальных уравнений вида (П 2.18) используются методы Эйлера, Рунге-Кутта и другие.

П 2.6.1. Метод Эйлера

Суть метода Эйлера заключается в аппроксимации интегральной кривой $y = F(x)$ ломаной линией. Метод Эйлера является одношаговым методом первого порядка.

Поиск решения производится на интервале $[x_0, a)$. На этом интервале вводится набор равноотстоящих точек $x_0 < x_1 < \ldots < x_n \le a$. Шаг разбиения $h = x_{i+1} - x_i$. Фактически необходимо получить таблицу значений функции $F(x)$ в каждой точке x_i.

При решении задачи методом Эйлера приближенное значение y_1 в точке x_1 с помощью уже известных значений x_0 и y_0 вычисляется по формуле

$$y_1 = y_0 + \Delta y_1,$$

где $\Delta y_1 = h \cdot f(x_0, y_0)$.

Значение функции $f(x_0, y_0)$ получается подстановкой x_0 и y_0 в функцию $f(x, y)$. Аналогично вычисляются значения во всех последующих точках интервала.

Таким образом, приближенное значение функции y_{i+1} в следующей точке рассматриваемого отрезка $[x_0, a)$ равно сумме значения функции y_i в предыдущей точке и произведения шага h на величину производной, вычисленной в предыдущей точке. То есть приближенные значения y_{i+1} в точках $x_{i+1} = x_0 + (i+1)h$, $i = 1, 2, ..., n$, вычисляются по формуле

$$y_{i+1} = y_i + \Delta y_i, \qquad (П\ 2.21)$$

где $\Delta y_i = h \cdot f(x_i, y_i)$.

Особенностью любого пошагового метода является то, что, начиная со второго шага, значение y_i является приближенным и поэтому погрешность на каждом следующем шаге возрастает. Приня-

то считать, что погрешность метода Эйлера пропорциональна величине h^2.

Пример решения дифференциального уравнения методом Эйлера

Постановка задачи. Решить дифференциальное уравнение $y' = 2y + e^x$ на отрезке [0, 2] при начальном условии $y(0) = 1$ методом Эйлера. Задать шаг разбиения отрезка $h = 0{,}5$.

Решение

Для решения уравнения используем итерационную формулу (П 2.21):

$$y_{i+1} = y_i + h(2y + e^x).$$

Отрезок [0, 2] разбивается с помощью шага $h = 0{,}5$ и формируется таблица значений функции $F(x)$ в каждой точке x_i ($x_0 = 0$; $x_1 = 0{,}5$; $x_2 = 1$; $x_3 = 1{,}5$; $x_4 = 2$) (табл. П2.1).

Таблица П 2.1

i	x_i	y_i
0	0	1
1	0,5	2,5
2	1	5,82
3	1,5	13,00
4	2	28,24

В точке $x_0 = 0$ значение $y_0 = 1$ согласно начальному условию.

В точке $x_1 = 0{,}5$ значение

$$y_1 = y_0 + 0{,}5 \cdot (2 \cdot y_0 + e^0) = 1 + 0{,}5(2 \cdot 1 + 1) = 2{,}5.$$

В точке $x_2 = 1$ значение $y_2 = y_1 + 0{,}5 \cdot (2 \cdot y_1 + e^{0{,}5}) =$

$$= 2{,}5 + 0{,}5(2 \cdot 2{,}5 + 1{,}65) = 5{,}82.$$

В точке $x_3 = 1{,}5$ значение $y_3 = y_2 + 0{,}5 \cdot (2 \cdot y_2 + e^1) =$

$$= 5{,}82 + 0{,}5(2 \cdot 5{,}82 + 2{,}72) = 13{,}00.$$

В точке $x_4 = 2$ значение $y_4 = y_3 + 0{,}5 \cdot (2 \cdot y_3 + e^{1{,}5}) = 13{,}00 + 0{,}5(2 \cdot 13{,}00 + 4{,}48) = 28{,}24$.

П 2.6.2. Метод Рунге-Кутта

Более эффективным методом для решения дифференциальных уравнений является метод Рунге-Кутта. В метод Рунге-Кутта входит семейство формул, которые отличаются способом вычисления приращения функции в точках разбиения x_i. Наиболее популярным является метод Рунге-Кутта четвертого порядка. Суть метода заключается в использовании на каждом шаге вычислений значений функции $f(x, y)$ в нескольких точках.

Аналогично методу Эйлера необходимо получить таблицу значений функции $F(x)$ в каждой точке x_i на интервале $[x_0, a)$. На этом интервале вводится набор равноотстоящих точек $x_0 < x_1 < \ldots < x_n \le a$. Шаг разбиения $h = x_{i+1} - x_i$.

В основе метода Рунге-Кутта, так же как и в методе Эйлера, лежит итерационная формула

$$y_{i+1} = y_i + \Delta y_i, \qquad (\text{П } 2.22)$$

где $\Delta y_i = \frac{h}{6} \cdot \left(k_1^{(i)} + 2k_2^{(i)} + 2k_3^{(i)} + k_4^{(i)}\right)$;

$k_1^{(i)} = h \cdot f(x_i, y_i)$;

$k_2^{(i)} = h \cdot f\left(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}\right)$;

$k_3^{(i)} = f\left(x_i + \frac{h}{2}, y_i + \frac{hk_2}{2}\right)$;

$k_4^{(i)} = f(x_i + h, y_i + hk_3)$.

Погрешность метода Рунге-Кутта пропорциональна величине h^5.

Пример решения дифференциального уравнения первого порядка методом Рунге-Кутта

Постановка задачи. Решить дифференциальное уравнение $y' = 2y + e^x$ на отрезке [0, 2] с начальными условиями $y(0) = 1$ методом Рунге-Кутта четвертого порядка. Задать шаг разбиения отрезка $h = 0{,}5$.

Пошаговый алгоритм решения

Для решения уравнения используем итерационную формулу (П 2.22). Отрезок [0, 2] разбивается с помощью шага $h = 0{,}5$ и формируется таблица (табл. П 2.2) для получения значений функции $F(x)$ в каждой точке x_i ($x_0 = 0$; $x_1 = 0{,}5$; $x_2 = 1$; $x_3 = 1{,}5$; $x_4 = 2$).

Шаг 0. В точке $x_0 = 0$ значение $y_0 = 1$ согласно начальному условию.

Шаг 1. В точке $x_1 = 0{,}5$ определяется значение y_1.

$$k_1 = f(x_0, y_0) = 2 \cdot y_0 + e^{x_0} = 2 \cdot 1 + e^0 = 3;$$

$$k_2 = f\left(x_0 + \frac{h}{2}, y_0 + h \cdot \frac{k_1}{2}\right) = 2 \cdot \left(y_0 + h \cdot \frac{k_1}{2}\right) + e^{x_0 + \frac{h}{2}} =$$

$$= 2 \cdot \left(1 + 0{,}5 \cdot \frac{3}{2}\right) + e^{0 + \frac{0{,}5}{2}} = 4{,}78;$$

$$k_3 = f\left(x_0 + \frac{h}{2}, y_0 + \frac{hk_2}{2}\right) = 2 \cdot \left(y_0 + h \cdot \frac{k_2}{2}\right) + e^{x_0 + \frac{h}{2}} =$$

$$= 2 \cdot \left(1 + 0{,}5 \cdot \frac{4{,}78}{2}\right) + e^{0 + \frac{0{,}5}{2}} = 5{,}67;$$

$$k_4 = f(x_0 + h, y_0 + hk_3) = 2 \cdot (y_0 + h \cdot k_3) + e^{x_0 + h} =$$

$$= 2 \cdot (1 + 0{,}5 \cdot 5{,}67) + e^{0 + 0{,}5} = 9{,}32;$$

$$y_1 = y_0 + \frac{h}{6} \cdot \left(k_1 + 2k_2 + 2k_3 + k_4\right) =$$

$$= 1 + 0{,}5 \cdot \frac{1}{6}\left(3 + 2 \cdot 4{,}78 + 2 \cdot 5{,}67 + 9{,}32\right) = 3{,}77.$$

Шаг 2. В точке $x_2 = 1$ определение значение y_2.

$$k_1 = f(x_1, y_1) = 2 \cdot y_1 + e^{x_1} = 2 \cdot 3{,}77 + e^{0{,}5} = 9{,}19;$$

$$k_2 = f(x_1 + \frac{h}{2}, y_1 + \frac{hk_1}{2}) = 2 \cdot (3{,}77 + 0{,}5 \cdot \frac{9{,}19}{2}) + e^{0{,}5 + \frac{0{,}5}{2}} = 14{,}25;$$

$$k_3 = f(x_1 + \frac{h}{2}, y_1 + \frac{hk_2}{2}) = 2 \cdot (3{,}77 + 0{,}5 \cdot \frac{14{,}25}{2}) + e^{0{,}5 + \frac{0{,}5}{2}} = 16{,}78;$$

$$k_4 = f(x_1 + h, y_1 + hk_3) = 2 \cdot (3{,}77 + 0{,}5 \cdot 16{,}78) + e^{0{,}5+0{,}5} = 27{,}03;$$

$$y_2 = y_1 + \frac{h}{6} \cdot (k_1 + 2k_2 + 2k_3 + k_4) =$$

$$= 3{,}77 + \frac{0{,}5}{6} (9{,}19 + 2 \cdot 14{,}25 + 2 \cdot 16{,}78 + 27{,}03) = 11{,}96.$$

Шаг 3. В точке $x_3 = 1{,}5$ определение значение y_3.

$$k_1 = f(x_2, y_2) = 2 \cdot y_2 + e^{x_2} = 2 \cdot 11{,}96 + e^1 = 26{,}63;$$

$$k_2 = f(x_2 + \frac{h}{2}, y_2 + \frac{hk_1}{2}) = 2 \cdot (11{,}96 + 0{,}5 \cdot \frac{26{,}63}{2}) + e^{1 + \frac{0{,}5}{2}} = 40{,}72;$$

$$k_3 = f(x_2 + \frac{h}{2}, y_2 + \frac{hk_2}{2}) = 2 \cdot (11{,}96 + 0{,}5 \cdot \frac{40{,}72}{2}) + e^{1 + \frac{0{,}5}{2}} = 47{,}76;$$

$$k_4 = f(x_2 + h, y_2 + hk_3) = 2 \cdot (11{,}96 + 0{,}5 \cdot 47{,}76) + e^{1+0{,}5} = 76{,}16;$$

$$y_3 = y_2 + \frac{h}{6} \cdot (k_1 + 2k_2 + 2k_3 + k_4) =$$

$$= 11{,}96 + \frac{0{,}5}{6} (6{,}63 + 2 \cdot 40{,}72 + 2 \cdot 47{,}76 + 76{,}16) = 36{,}27.$$

Шаг 4. В точке $x_4 = 2$ определение значение y_4.

$$k_1 = f(x_3, y_3) = 2 \cdot y_3 + e^{x3} = 2 \cdot 36{,}27 + e^{1{,}5} = 75{,}02;$$

$$k_2 = f(x_3 + \frac{h}{2}, y_3 + \frac{hk_1}{2}) = 2 \cdot (36{,}27 + 0{,}5 \cdot \frac{75{,}02}{2}) + e^{1{,}5 + \frac{0{,}5}{2}} = 113{,}81;$$

$$k_3 = f(x_3 + \frac{h}{2}, y_3 + \frac{hk_2}{2}) = 2 \cdot (36{,}27 + 0{,}5 \cdot \frac{113{,}81}{2}) + e^{1{,}5 + \frac{0{,}5}{2}} = 133{,}20;$$

$$k_4 = f(x_3 + h, y_3 + hk_3) = 2 \cdot (36{,}27 + 0{,}5 \cdot 113{,}20) + e^{1{,}5+0{,}5} = 211{,}13;$$

$$y_4 = y_3 + \frac{h}{6}\cdot(k_1 + 2k_2 + 2k_3 + k_4) = 36{,}27 + \frac{0{,}5}{6}(75{,}02 +$$
$$+ 2\cdot 113{,}81 + 2\cdot 133{,}20 + 211{,}13) = 100{,}28.$$

Результаты вычислений приведены в табл. П2.2.

Таблица П 2.2

i	x_i	y_i
0	0	1
1	0,5	3,77
2	1	11,96
3	1,5	36,27
4	2	100,28

Решение дифференциальных уравнений порядка $f(x, y, y', y'', ..., y^{(n)})$

Для решения дифференциального уравнения более высокого порядка $f(x, y, y', y'', ..., y^{(n)})$ используется следующий прием. Любое уравнение n-го порядка можно свести к системе n уравнений первого порядка способом замены переменных. Для этого вводят новые переменные $y' = y_1, y'' = y_2, ..., y^{(n-1)} = y_{n-1}$, в результате чего получают эквивалентную систему

$$\begin{cases} y' = y_1, \\ y_1' = y_2, \\ \ldots \\ y_{n-1}' = f(x, y, y_1, y_2, ..., y_{n-1}). \end{cases}$$

Указанный метод позволяет свести решение дифференциального уравнения n-го порядка к решению системы n уравнений первого порядка. В свою очередь, методы решения одного уравнения первого порядка распространяются на систему таких уравнений.

Пример решения дифференциального уравнения второго порядка методом Рунге-Кутта четвертого порядка

Постановка задачи. Решить дифференциальное уравнение $y''-3y'+y+4x=0$ с начальными условиями $y(1)=5$, $y'(1)=-1$ на отрезке [1, 3] методом Рунге-Кутта четвертого порядка.

Решение

Выполняя замену переменной $y'=y_1$, приходим к системе из двух уравнений:

$$\begin{cases} y'=y_1, \\ y_1'=3y_1-y-4x. \end{cases}$$

Таким образом, любое уравнение n-го порядка можно свести к системе уравнений первого порядка.

П 2.7. Основные теоретические сведения о численных методах решения дифференциальных уравнений в частных производных

Многие математические модели описываются дифференциальным уравнением или системой дифференциальных уравнений с краевыми условиями первого, второго или третьего рода. Точное решение краевых задач удается получить лишь для немногих частных случаев. Поэтому общий способ решения таких дифференциальных уравнений заключается в использовании различных приближенных моделей. Наиболее распространены модели на основе метода конечных разностей (метод сеток).

Сущность метода состоит в том, что область изменения независимых переменных (накладывается сетка, т. е. область разбивается на определенные участки в виде отрезков, прямоугольников, параллелепипедов в зависимости от размерности модели. Так, для одномерной модели сетка может иметь вид, представленный на рис. П 2.5.

Рис. П 2.5

Рис. П 2.5

Для двумерных задач наиболее часто применяется прямоугольная сетка, узлы которой лежат на пересечении прямых, параллельных координатным осям (рис. П 2.6), для трехмерных – сетка из прямоугольных параллелепипедов, узлы которой лежат на пересечении плоскостей, параллельных координатным плоскостям (рис. П 2.7). Расстояния между узлами называются шагом сетки по соответствующей переменной (Δx, Δy, Δz).

Для двумерных задач наиболее часто применяется прямоугольная сетка, узлы которой лежат на пересечении прямых, параллельных координатным осям (рис. П 2.6), для трехмерных – сетка из прямоугольных параллелепипедов, узлы которой лежат на пересечении плоскостей, параллельных координатным плоскостям (рис. П 2.7). Расстояния между узлами называются шагом сетки по соответствующей переменной (Δx, Δy, Δz).

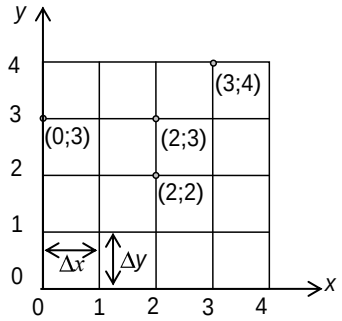


Рис. П 2.6

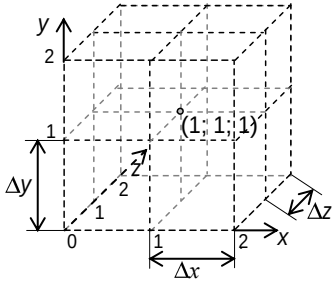


Рис. П 2.7

Так как математические модели обычно описывают физические или химические процессы, протекающие в течение определенного промежутка времени, то весь временной интервал также разбивается с некоторым шагом t на равномерные малые отрезки, называемые временными слоями. Вычисления искомой функции производятся последовательно на каждом временном слое – от начального момента времени до конечного.

Так как математические модели обычно описывают физические или химические процессы, протекающие в течение определенного промежутка времени, то весь временной интервал также разбивается с некоторым шагом Δt на равномерные малые отрезки, называемые временными слоями. Вычисления искомой функции производятся последовательно на каждом временном слое – от начального момента времени до конечного.

Таким образом область непрерывного изменения аргументов заменяется расчетной сеткой – дискретным множеством точек (узлов). Вместо функции непрерывных аргументов вводятся функции дискретных аргументов, определяемые в узлах сетки. Например, при построении одномерной модели, описывающей нагрев стержня длиной 10 мм, вместо области непрерывного изменения переменной x от 0 до 10 мм рассматриваются дискретные значения x_i = 0; 0,5; 1; …; 9,5; 10 (шаг сетки задан как Δx = 0,5). Следует отметить, что номера узлов в данном случае изменяются как 0, 1, 2, …, 21. Соот-

ветственно решением дифференциального уравнения является не непрерывная функции $T(x, t)$, а ее дискретный аналог T_i^j, где i – координата узла для переменной x; j – номер временного слоя.

Таким образом, метод конечных разностей состоит в замене (аппроксимации) всех частных производных дифференциального уравнения разностными соотношениями, а именно приближенным вычислением бесконечно малых приращений функций и аргументов, которые называются конечными разностями. Все производные, входящие в уравнение, заменяют разностями значений функции в ближайших узлах пространственной сетки. Так, например, для двумерной сетки вычисление функции для узла (i, j) производится с использованием значений функций, полученных в узлах (i, j + 1), (i, j – 1), (i + 1, j), (i – 1, j) (рис. П 2.8).

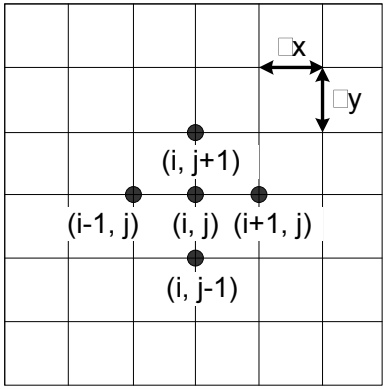


Рис. П 2.8

Применение этого метода позволяет свести дифференциальную краевую задачу к системе нелинейных, а в общем случае – алгебраических уравнений относительно неизвестных значений функций в узлах пространственной сетки.

Решение начинается с первого временного слоя, поскольку нулевой слой по времени определяется начальными условиями. Значения функции в каждом из узлов пространственной сетки на первом временном слое определяются по ее значениям в ближайших узлах нулевого слоя. После того как будут известны значения функции в узлах пространственной сети первого временного слоя, аналогично

рассчитываются значения функции в узлах второго слоя, затем – третьего и т. д.

Метод конечных разностей для решения двумерной стационарной задачи

Постановка задачи. Пусть дана прямоугольная металлическая пластина с известными значениями теплопроводности (λ), теплоемкости (c) и плотности (ρ), начальная температура которой равна T_0. Она помещается в температурное поле с температурой T_1. Необходимо рассчитать тепловое поле пластины (значение температуры в каждой точке) в последующие моменты времени.

Температура любой точки пластины зависит не только от расположения точки (координат x, y), но и от времени (t). Очевидно, что с течением времени все области пластины нагреваются сильнее и тем больше, чем ближе расположены к границе пластины. Известно, что изменение температуры подчиняется уравнению в частных производных

$$\frac{\partial T}{\partial t} = a\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right), \qquad (\text{П } 2.23)$$

где $T = f(x, y, t)$ – температура пластины в точке с координатами (x, y) в момент времени t;

a – коэффициент температуропроводности материала:

$$a = \frac{\lambda}{c \cdot \rho},$$

где λ – теплопроводность.

Решение

Шаг 1. Построение сетки в заданной области. Для решения уравнения (П 2.23) численным методом необходимо построить сетку с пространственным шагом $h = \Delta x = \Delta y$ (рис. П 2.9). При этом пластина разбивается на большое количество узлов, находящихся на пересечении вертикальных и горизонтальных линий. Каждый узел задается парой чисел (i, j), где $i = 0; n$, $j = 0; m$. Таким образом, от непрерывных переменных x и y переходим к дискретным набо-

разбивается на большое количество узлов, находящихся на пересечении вертикальных и горизонтальных линий. Каждый узел задается парой чисел (i, j), где $i = 0, n$, $j = 0, m$. Таким образом, от непрерывных переменных x и y переходим к дискретным наборам значений $x_1, x_2, \ldots, x_{m-1}, x_m$ и $y_1, y_2, \ldots, y_{n-1}, y_n$. От непрерывного времени t переходим к дискретному $t_k = k$, $k = 0, 1, 2, \ldots, l$ – временные слои.

рам значений $x_1, x_2, \ldots, x_{m-1}, x_m$ и $y_1, y_2, \ldots, y_{n-1}, y_n$. От непрерывного времени t переходим к дискретному t_k так, что $t_k = k\Delta t$, $k = 0, 1, 2, \ldots, l$ – временные слои.

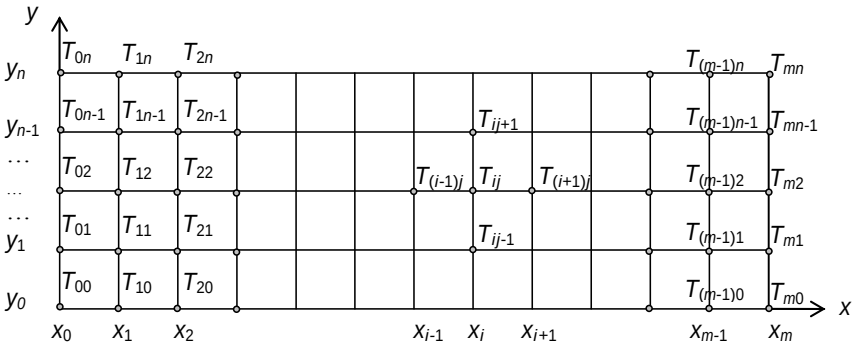


Рис. П 2.9

Решением исходной задачи является определение температуры в последующие моменты времени в каждом узле сетки, т.е. в узлах $T_{00}, T_{01}, T_{0n-1}, T_{0n}, T_{10}, T_{11}, T_{1n-1}, T_{1n}, T_{m0}, T_{m1}, T_{mn-1}, T_{mn}$.

Шаг 2. Задание начального условия. Поскольку в начальный момент времени внутри пластины температура точек равна T_0, начальное условие имеет вид

$$T_{ij}^0 = T_0, \qquad (\text{П } 2.24)$$

где T_{ij}^0 – температура в узле (i, j) плоскости $x0y$ на нулевом временном слое, где $i = 0; n$, $j = 0; m$.

Шаг 3. Задание граничного условия. Поскольку в начальный момент крайние точки пластины имеют температуру T_1 (так как пластина была помещена в температурное поле с температурой T_1), то граничное условие (условие для узлов пластины, находящихся на границе) имеет вид

$$T_{i_1, j_1}(t = 0) = T_1, \qquad (\text{П } 2.25)$$

где T_{i_1,j_1} – температура в точке с координатами i_1, j_1 по осям x, y соответственно, где $i=0$ и $j=\overline{0,m}$ или $i=\overline{0,n}$ и $j=0$.

Таким образом, используя начальное и граничное условие можно записать формулу определения температуры пластины в начальный момент времени ($t = 0$):

$$T_{i,j}^0=\begin{cases}T_0, \text{ если } (i=0 \text{ и } j=\overline{0,m}) \text{ или } (i=\overline{0,n} \text{ и } j=0);\\ T_1, \text{ для остальных точек.}\end{cases}$$

Шаг 4. Замена дифференциальных операторов $\frac{\partial T}{\partial t}$, $\frac{\partial^2 T}{\partial x^2}$, $\frac{\partial^2 T}{\partial y^2}$ в исходном дифференциальном уравнении разностными аналогами, построенными с помощью формулы конечных разностей. Так, первая производная заменяется конечными разностями с использованием формулы (П 2.11):

$$\frac{\partial T}{\partial t}=\frac{T_{i,j}^k-T_{i,j}^{k-1}}{\Delta t}. \qquad \text{(П 2.26)}$$

Вторые производные заменяются вычислением конечных разностей по формуле (П 2.14):

$$\frac{\partial^2 T}{\partial x^2}=\frac{T_{i+1,j}^{k-1}-2T_{i,j}^{k-1}+T_{i-1,j}^{k-1}}{\Delta x^2};$$

$$\frac{\partial^2 T}{\partial y^2}=\frac{T_{i,j+1}^{k-1}-2T_{i,j}^{k-1}+T_{i,j-1}^{k-1}}{\Delta y^2}, \qquad \text{(П 2.27)}$$

где $T_{i,j}$ – температура в узле (i, j) плоскости $x0y$;

k – номер временного слоя;

Δx и Δy – шаги сетки для величин x и y.

Таким образом, уравнение (П 2.23) после подстановки формул (П2.24)–(П2.27) будет иметь вид

$$T_{i,j}^{k} = T_{i,j}^{k-1} + a\Delta t\left(\frac{T_{i+1,j}^{k-1} - 2T_{i,j}^{k-1} + T_{i-1,j}^{k-1}}{\Delta x^2} + \frac{T_{i,j+1}^{k-1} - 2T_{i,j}^{k-1} + T_{i,j-1}^{k-1}}{\Delta y^2}\right). \quad (\text{П } 2.28)$$

Шаг 5. Составление системы алгебраических уравнений:

$$\begin{cases} T_{i,j}^{0} = \begin{cases} T_0, \text{ если } (i = 0 \text{ и } j = \overline{0,m}) \text{ или } (i = \overline{0,n} \text{ и } j = 0); \\ T_1 \text{ для остальных точек;} \end{cases} \\ T_{i,j}^{1} = T_{i,j}^{0} + a\Delta t\left(\dfrac{T_{i+1,j}^{0} - 2T_{i,j}^{0} + T_{i-1,j}^{0}}{\Delta x^2} + \dfrac{T_{i,j+1}^{0} - 2T_{i,j}^{0} + T_{i,j-1}^{0}}{\Delta y^2}\right); \\ T_{i,j}^{2} = T_{i,j}^{1} + a\Delta t\left(\dfrac{T_{i+1,j}^{1} - 2T_{i,j}^{1} + T_{i-1,j}^{1}}{\Delta x^2} + \dfrac{T_{i,j+1}^{1} - 2T_{i,j}^{1} + T_{i,j-1}^{1}}{\Delta y^2}\right); \\ \ldots \\ T_{i,j}^{l} = T_{i,j}^{l-1} + a\Delta t\left(\dfrac{T_{i+1,j}^{l-1} - 2T_{i,j}^{l-1} + T_{i-1,j}^{l-1}}{\Delta x^2} + \dfrac{T_{i,j+1}^{l-1} - 2T_{i,j}^{l-1} + T_{i,j-1}^{l-1}}{\Delta y^2}\right). \end{cases}$$

Шаг 6. Решение полученной системы. Решение системы можно осуществлять поэтапно, т. е. сначала в момент времени t_1 вычислять значения $T_{i,j}^{1}$ по известным значениям $T_{i,j}^{0}$, затем в момент времени t_2 – значения $T_{i,j}^{2}$ и так до конечного момента времени τ_l – значения $T_{i,j}^{l}$.

Учебное издание

ЧИЧКО Александр Николаевич
САЧЕК Ольга Александровна
ЧИЧКО Ольга Ильинична

ИНФОРМАТИКА. ПРАКТИКУМ

Учебное пособие

Редактор Т.Н. Микулик
Компьютерная верстка Д.А. Исаева

Подписано в печать 07.07.2011.
Формат 60×84 $^{1}/_{16}$. Бумага офсетная.
Отпечатано на ризографе. Гарнитура Таймс.
Усл. печ. л. 23,19. Уч.-изд. л. 18,14. Тираж 100. Заказ 1339.

Издатель и полиграфическое исполнение:
Белорусский национальный технический университет.
ЛИ № 02330/0494349 от 16.03.2009.
Проспект Независимости, 65. 220013, Минск.