МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе
по дисциплине «Основы алгоритмизации и программирования»

на тему: «*Программирование с использованием подпрограмм на языке С*»

Исполнитель: студент гр. ИТ-11
Ю.В. Вегеро
Руководитель: преподаватель
Иванов И.И.

Дата проверки: _____
Дата допуска к защите: _____
Дата защиты: _____

Оценка работы: _____

Подписи членов комиссии
по защите курсовой работы: _____

Гомель 2011

Содержание

Введение

Объектно-ориентированный подход к проектированию больших программных систем в последнее десятилетие стал безусловным стандартом, поскольку способствует лучшей управляемости проектом на всех стадиях разработки, тестирования и внедрения. Соответственно, всё большее развитие получают языки программирования с объектно-ориентированными возможностями. Одним из наиболее распространённых объектно-ориентированных языков является C++, созданный Бьерном Страуструпом. Этот язык обладает настолько большой гибкостью и широкими возможностями, что, если не поставить себя в жесткие рамки с самого начала, программа быстро превратится в огромного неуправляемого монстра, не поддающегося отладке.

В C++ не существует единственного самого лучшего способа создания программ. Для решения задач разного рода и уровня сложности требуется применять разные технологии программирования. В простейших случаях достаточно освоить азы структурного написания программ. Для создания же сложных проектов требуется не только свободно владеть языком в полном объеме, но и иметь представление о принципах проектирования и отладки программ, возможностях стандартной и других библиотек и т. д.

При разработке больших и сложных алгоритмов и программ логически независимые или повторяющиеся последовательности действий оформляют в виде вспомогательных алгоритмов (для алгоритмов) и подпрограмм (для программ).

С увеличением объема программы становится невозможным удерживать в памяти все детали. Естественным способом борьбы со сложностью любой задачи является ее разбиение на части. В C++ задача может быть разделена на более простые и обозримые с помощью функций, после чего программу можно рассматривать в более укрупненном виде — на уровне взаимодействия функций. Это важно, поскольку человек способен помнить ограниченное количество фактов.

Функция – это логически самостоятельная часть программы, имеющая имя, которой могут передаваться параметры и которая может возвращать какое-то значение.

Можно представлять функцию как некоторую машину или «черный ящик», в который Вы вкладываете данные и из которого получаете результат. Внутренне устройство и способ действия функции невидимы и неизвестны для остальной части программы. Внешний мир должен знать о некоторой функции лишь только то, что в нее вводится, что из

нее выводится и какие необычные эффекты вызывает выполнение этой функции.

Использование функций является первым шагом к повышению степени абстракции программы и ведет к упрощению ее структуры. Разбив задачу на подзадачи и оформив каждую из них в виде функций, мы поступаем по принципу, известному еще с древних времен: ***"Разделяй и властвуй".***

Разделение программы на функции позволяет также избежать избыточности кода, поскольку функцию записывают один раз, а вызывать ее на выполнение можно многократно из разных точек программы. Процесс отладки программы, содержащей функции, можно лучше структурировать. Часто используемые функции можно помещать в библиотеки.

Библиотеки функций позволяют:

- использовать одни и те же функции разными программистами.
- повысить структурированность и наглядность программ.
- облегчить чтение, освоение и корректировку программы.

Таким образом, создаются более простые в отладке и сопровождении программы.

1. Постановка задач и исходные данные

1. Для решения каждой задачи в соответствии с условием, требуется разбить задачу на подзадачи и разработать вспомогательные и основной алгоритмы.
2. Оформить разработанные алгоритмы в виде графических схем.
3. Написать программу с использованием подпрограмм, соответствующую разработанным алгоритмам.
4. Отладить программу в среде программирования.
5. Каждая подпрограмма в качестве входных параметров должна иметь массив и количество его элементов (при наличии массива). Результат выполнения подпрограммы передавать через ее заголовок и/или по оператору return.
6. Исходные данные для отладки программы подобрать самостоятельно.
7. Подготовить полный набор тестов для отладки разработанных программ.

№	Задание	Используемые формулы
1	Вычислить площадь и периметр прямоугольника со сторонами a, b.	$S = ab$ $P = 2(a+b)$
2	Вычислить значение функции f, где sn, sm, s5 - суммы n элементов массива X, m элементов массива Y, 5 элементов массива Z, соответственно.	$f = \frac{\sin(sn) + \cos^2(sm)}{3{,}2\ s5}$
3	Решить уравнение $ax^2 + bx + c = 0$, где a, b, c - максимальные значения элементов матриц X, Y и Z, соответственно.	-
4	Разработать и оформить в виде графической схемы алгоритм вычисления определенного интеграла с точностью ξ методом центральных прямоугольников.	$f1=(x+6.25)/(x+1.5)^2$ $f2= x^2+5x\sin x-7$ $f3=(x^2-6x+1)/(x-3)$

Примечание: в задании 4 в функцию необходимо передать указатель на подынтегральную функцию.

2. Передача параметров в функцию

2.1 Теоретические сведения

Существует два способа передачи параметров в функцию: по значению и по адресу. При передаче по значению в стек заносятся копии значений аргументов, и операторы функции работают с этими копиями. Доступа к исходным значениям параметров у функции нет, а, следовательно, нет и возможности их изменить.
При передаче по адресу в стек заносятся копии адресов аргументов, а функция
осуществляет доступ к ячейкам памяти по этим адресам и может изменить исходные значения аргументов:

```
#include<stdio.h>
#include <iostream.h>
void f(int i, int* j , int& k);
int main()
{
  int i = 1, j = 2, k = 3;
   cout «"i j k\n";
   cout «  i «' '« j «' '« k «'\n';
  f(i, &j, k);
  cout «i«' '« j «' '« k;
  return (0);
}
void f(int i, int* j , int& k)
{
  i++; (*j)++; k++;
}
```

Результат работы программы:

```
i   j   k
1  2  3
1  3  4
```

Первый параметр (i) передается по значению. Его изменение в функции не влияет на исходное значение. Второй параметр (j) передается по адресу с помощью указателя, при этом для передачи в функцию адреса фактического параметра используется операция взятия адреса, а для получения его значения в функции требуется операция разыменования. Третий параметр (k) передается по адресу с помощью ссылки.

При передаче по ссылке в функцию передается адрес указанного при вызове параметра, а внутри функции все обращения к параметру неявно разыменовываются. Поэтому использование ссылок вместо указателей улучшает читаемость программы, избавляя от необходимости применять операции получения адреса и разыменования. Использование ссылок вместо передачи по значению более эффективно, поскольку не требует копирования параметров, что имеет значение при передаче структур данных большого объема.

2.2 Графическая схема алгоритма

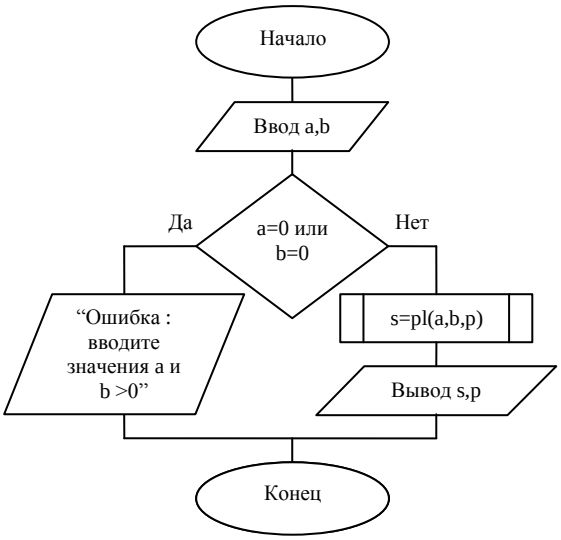


Рисунок 1 - Графическая схема основного алгоритма

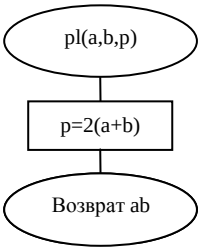


Рисунок 2 - Графическая схема алгоритма вычисления периметра и площади

2.3 Текст программы

```
#include <stdio.h> //Директивы подключения
#include<conio.h>  //файлов
float pl(float a,float b,float &p); //прототип функции
void main() // главная функция, не возвращающая значение
{
  clrscr();
```

```
 float a,b,       // стороны прямоугольника
       p,         // периметр прямоугольника
       s;       // влощадь прямоугольника
 printf("Введите значения a и b\n");
 scanf("%f%f",&a,&b); //ввод значений a и b
 if(a==0||b==0)       // условие выполнения вычислений
    printf("Ошибка : вводите значения a и b > 0\n");
 else
 {
    s=pl(a,b,p);    // присваиваем переменной s значение, возвращённое
                    // функцией по оператору return
    printf("Площадь прямоугольника равна %.3f, периметр равен
%.3f\n",s,p);
  }
 fflush(stdin);
 getch();
}

float pl(float a,float b,float &p) // подпрограмма вычисления площади и
                                   // периметра
{
  p=2*(a+b);       //вычисление периметра прямоугольника
  return (a*b);    //возврат площади прямоугольника
}
```

2.4 Тесты

1) Исходные данные: a=4, b=3 — Результат: площадь прямоугольника равна 12.000, периметр равен 14.000
2) Исходные данные: a=3.5, b=2 — Результат: площадь прямоугольника равна 7.000, периметр равен 11.000
3) Исходные данные: a=1, b=0 — Результат: Ошибка: вводите значения a и b > 0
4) Исходные данные: a=0, b=0 — Результат: Ошибка: вводите значения a и b > 0
5) Исходные данные: a=6.543, b=7.112 — Результат: площадь прямоугольника равна 46.534, периметр равен 27.310

3. Передача одномерных массивов в функцию

3.1 Теоретические сведения

При использовании в качестве параметра массива в функцию передается указатель на его первый элемент, иными словами, массив передается по адресу. При этом информация о количестве элементов массива теряется, и следует передавать его размерность через отдельный параметр (в случае массива символов, то есть строки, ее фактическую длину можно определить по положению нуль-символа).

Пример:

```
float Mas3(float *a, int n);
```

При передаче массива в функцию размер массива можно указать при описании параметра или описать параметр с пустыми скобками в заголовке функции. В последнем случае добавляется дополнительный параметр, через который передается в функцию количество элементов массива.

Пример:

```
float Mas (float V[15]);
float Mas1(float V[], int n);
```

3.2 Графическая схема алгоритма

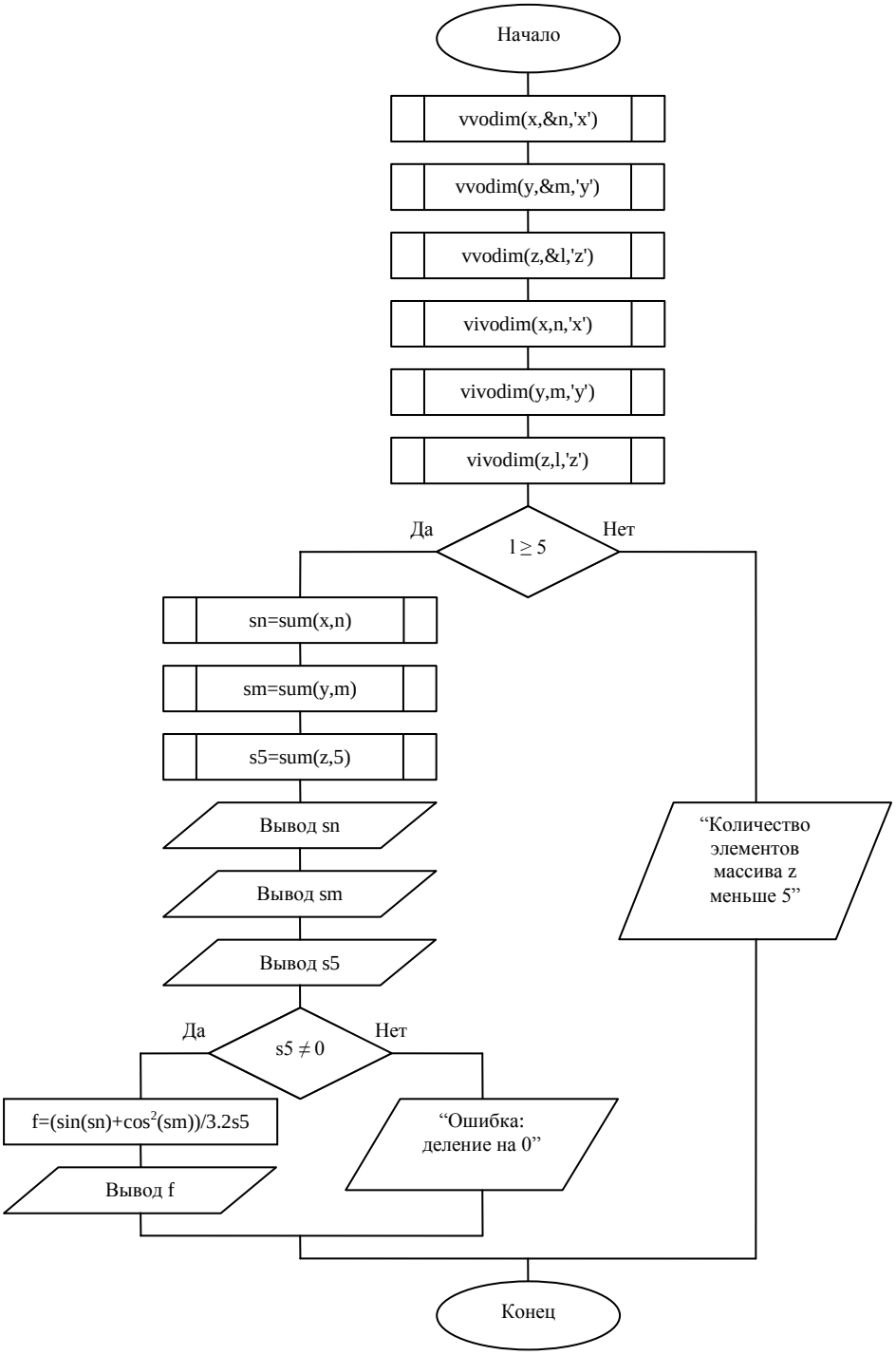


Рисунок 3 - Графическая схема основного алгоритма

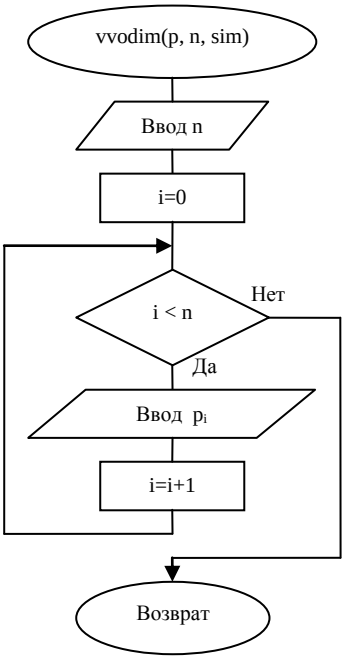


Рисунок 4 - Графическая схема алгоритма ввода массива

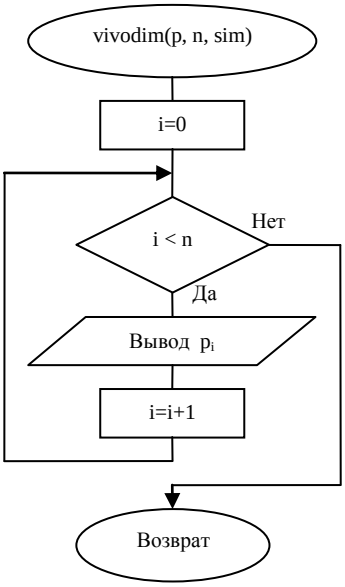


Рисунок 5 - Графическая схема алгоритма вывода массива

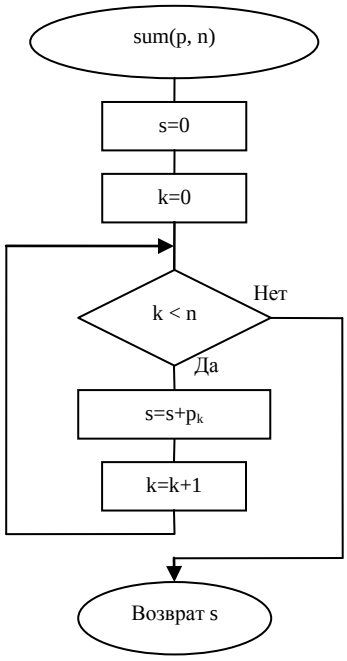


Рисунок 6 - Графическая схема алгоритма вычисления суммы элементов

3.3 Текст программы

```
#include<stdio.h>        // Директивы
#include<conio.h>        // подключения
#include<math.h>         // файлов
int sum(int *p,int n);     // прототип функции вычисления суммы
                           //элементов массива
void vvodim(int *p,int *n,char sim);   // прототип функции ввода массива
void vivodim(int *p,int n,char sim);   // прототип функции вывода массива
void main()                            // главная функция, не
                                       //возвращающая значения
{
  clrscr();
  float f;       // итоговое значение функции
  int x[50];    // описание
  int y[50];    // целочисленных
  int z[50];    // массивов
  int n,        // количество элементов массива x
     m,         // количество элементов массива y
     l,         // количество элементов массива z
     sn,        // сумма элементов массива x
     sm,        // сумма элементов массива y
     s5;        // сумма пяти элементов массива z
```

```
 vvodim(x,&n,'x');    // обращения к функции vvodim,
 vvodim(y,&m,'y');  //  не возвращающей значения
 vvodim(z,&l,'z');     // по оператору return
 vivodim(x,n,'x');     // обращения к функции vivodim,
 vivodim(y,m,'y');    //  не возвращающей значения
 vivodim(z,l,'z');      // по оператору return
 printf("\n");
 sn=sum(x,n);         // сумма элементов массива x
 sm=sum(y,m);        // сумма элементов массива y
 s5=sum(z,5);         // сумма пяти элементов массива z
 if(l>=5)              //  если кол-во элементов массива z  ≥ 5, то
                      //выполняются  вычисления
 {
   printf("\nСумма элементов массива x равна %d\n",sn);
   printf("\nСумма элементов массива y равна %d\n",sm);
   printf("\nСумма пяти элементов массива z равна %d\n",s5);
   if(s5!=0)         //проверка знаменателя  искомой функции на наличие 0
   {
      f=(sin(sn)+cos(sm)*cos(sm))/(3.2*s5);              // вычисление
                                                         // значения функции
      printf("\nЗначение функции равно %.4f\n",f);    // вывод значения
                                                         //функции
   }
     else printf("\nОшибка: деление на 0\n");           // вывод сообщения в
                                                         //случае ошибки
 }
 else           //   если кол-во элементов массива z < 5, то вывод сообщения
        printf("\nКоличество элементов массива z меньше 5!\n");
 fflush(stdin); getchar();
}
void vvodim(int *p,int *n,char sim)       // функция ввода массива, не
                                          // возвращающая
                                          //значения
{
  int i;                                  // целочисленный параментр цикла
  printf("Введите размерность массива %c  ",sim);
  scanf("%d",n);                          // ввод размерности массива
  for(i=0;i<*n;i++)
   {
     printf("%c[%d]=",sim,i);
     scanf("%d",&p[i]);                   // ввод элементов массива
   }
```

```
}
void vivodim(int *p,int n,char sim)  // функция вывода массива, не
                                     // возвращающая значения
{
 int i;                              // целочисленный параметр цикла
 printf("\n\nМассив %c\n\n",sim);
 for(i=0;i<n;i++)
  {
   printf(" %d ",p[i]);              // вывод элементов массива
  }
}
int sum(int *p,int n)       // функция вычисления суммы элементов массива
{
   int s=0,                 // сумма элементов массива
       k;                   // параметр цикла
   for(k=0;k<n;k++)         // цикл вычисления суммы элементов массива
     s+=*(p+k);             // вычисление суммы элементов массива
 return(s);                 //возврат  суммы элементов массива
}
```

3.4 Тесты

1) Исходные данные:
n=5 m=4 l=7
x={1,2,3,4,5}
y={2,5,3,1}
z={1,2,5,2,3,4,5}

Результат:
Сумма элементов массива x равна 15
Сумма элементов массива y равна 11
Сумма пяти элементов массива z равна 13
Значение функции равно 0.0156

2) Исходные данные:
n=3 m=4 l=3
x={3,4,5}
y={1,1,1,1}
z={1,2,5}

Результат:
Количество элементов массива z меньше 5 !

3) Исходные данные:
n=5 m=4 l=6
x={-3,4,-5,1,1}
y={1,0,0,1}
z={-1,2,-5,4,0,0}

Результат:
Сумма элементов массива x равна -2
Сумма элементов массива y равна 2
Сумма пяти элементов массива z равна 0
Ошибка: деление на 0

4) Исходные данные:
n=1 m=1 l=4
x={3}
y={1}
z={0,0,0,0}

Результат:
Количество элементов массива z меньше 5 !

4. Передача двумерных массивов в функцию

4.1 Теоретические сведения

Существует несколько способов решения проблемы передачи в качестве параметров многомерных массивов.

Первый путь. Описание соответствующего параметра должно в явном виде указывать все размерности, кроме старшей, например,

int b[][40][20].

Второй путь. Замена многомерного массива одномерным и имитация доступа к многомерному массиву внутри функции.

Третий путь. Использование вспомогательных массивов указателей на массивы.

Четвертый путь. Использование классов для представления массивов.

4.2 Графическая схема алгоритма

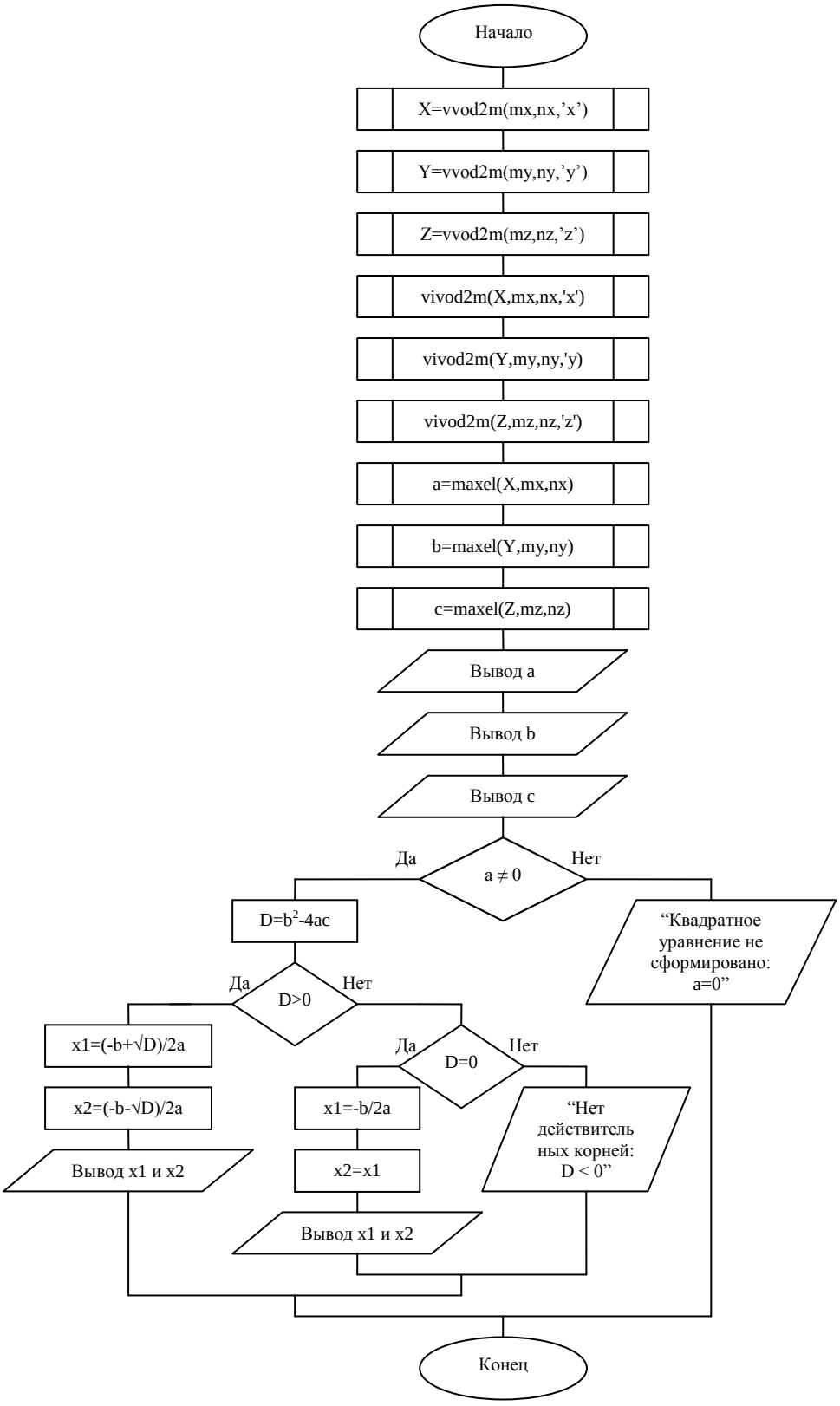


Рисунок 7 - Графическая схема основного алгоритма

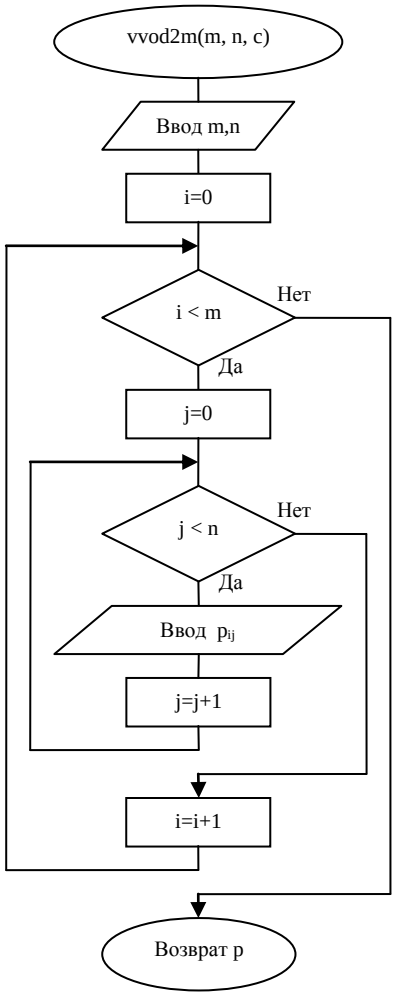


Рисунок 8 - Графическая схема алгоритма ввода матрицы

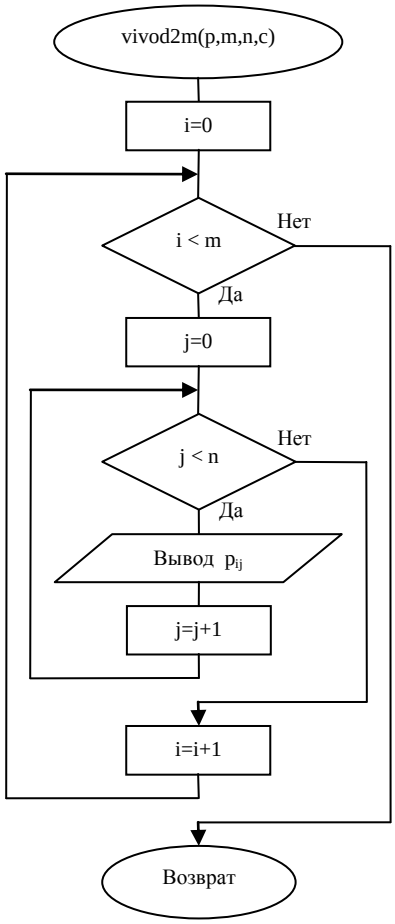


Рисунок 9 - Графическая схема алгоритма вывода матрицы

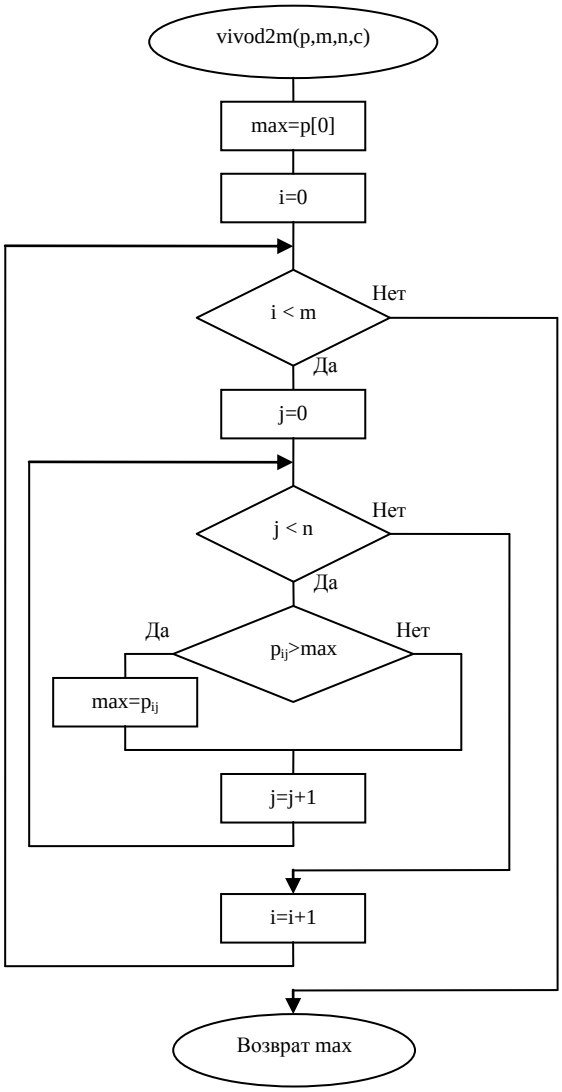


Рисунок 10 - Графическая схема алгоритма поиска максимального элемента

4.3 Текст программы

```
#include <stdio.h>   // Директивы
#include<conio.h>    // подключения
#include<math.h>     // файлов
int *vvod2m ( int *m, int *n, char c);       // прототип функции ввода
                                              //массива
void vivod2m(int *p, int m, int n, char c);  // прототип функции вывода
                                              // массива
int maxel(int *p, int m, int n);             // прототип функции, вычисляющей
                                             // макс. элемент массива
void main(void)                              // главная функция, не возвращающая
                                            // значения
```

```
{
 clrscr();
 int *X,*Y,*Z;   // целочисленные массивы
 int mx,             // количество строк массива X
    nx,              // количество столбцов массива X
    my,              // количество строк массива Y
    ny,              // количество столбцов массива Y
    mz,              // количество строк массива Z
    nz;              // количество столбцов массива Z
 int  a,             // максимальный элемент массива X
     b,              // максимальный элемент массива Y
     c,              // максимальный элемент массива Z
     D;              // дискриминант уравнения
 float x1,          // 1-ый корень квадратного уравнения
      x2;           // 2-ой корень квадратного уравнения
 X=vvod2m(&mx,&nx,'X');
 Y=vvod2m(&my,&ny,'Y');
 Z=vvod2m(&mz,&nz,'Z');
 vivod2m(X,mx,nx, 'X'); // вывод массива X
 vivod2m(Y,my,ny,'Y');  // вывод массива Y
 vivod2m(Z,mz,nz,'Z');  // вывод массива Z
 a=maxel(X,mx,nx);
 b=maxel(Y,my,ny);
 c=maxel(Z,mz,nz);
 printf("\nМаксимальный элемент массива X равен %d\n ",a);   // вывод
                                          // максимального элемента массива X
 printf("\nМаксимальный элемент массива Y равен %d\n ",b);     // вывод
                                          // максимального элемента массива Y
 printf("\nМаксимальный элемент массива Z равен %d\n ",c);      // вывод
                                          // максимального элемента массива Z
 if(a!=0)         // условие вычисления дискриминанта и корней уравнения
 {
   D=b*b-4*a*c;         // вычисление дискриминанта
   if(D>0)              // условие вычисления корней уравнения
   {
     x1=(-b+sqrt(D))/(2*a);  // вычисление первого корня уравнения
     x2=(-b-sqrt(D))/(2*a);  // вычисление второго корня уравнения
     printf("\nКорни уравнения: %.3f и %.3f\n",x1,x2);  // вывод корней
                                                       // уравнения
   }
  else
```

```
     if(D==0)
       {
         x1=-b/(2*a);   // вычисление корня уравнения, если дискриминант
                        // равен 0
         x2=x1;         // присваивание значения первого корня второму
                        // корню уравнения
         printf("\nКорни уравнения: %.3f и %.3f\n",x1,x2);  // вывод корней
                        //уравнения
       }
      else puts("\nНет действительных корней: дискриминант меньше 0");
      // вывод сообщения, если дискриминант меньше 0
   }
   else printf("\nКвадратное уравнение не сформировано: a=0\n"); // вывод
                  // сообщения в  случае, если уравнение   не сформировано
  delete[] X;      //освобождение динам. памяти, выделенной под массив X
  delete[] Y;     //освобождение динам. памяти, выделенной под массив Y
  delete[] Z;      //освобождение динам. памяти, выделенной под массив Z
  getch();
  fflush(stdin);
}
 /* Описание функций */
int *vvod2m(int *m, int *n, char c)  // функция ввода элементов массива
{
    int mm,    // кол-во элементов массива
        i,     // параметр цикла
        j,     // параметр цикла
       *p;     // массив
    printf("Введите количество строк и столбцов массива  %c\n",c);
    scanf("%d %d",m,n); // ввод кол-ва строк и столбцов массива
    mm=*m**n;           // вычисление кол-ва элементв массива
    p=new int [mm];     // выделение динамической памяти под массив
    for(i=0; i<*m; i++)    // внешний цикл ввода элементов массива
      for(j=0;j<*n;j++)    // внутренний цикл ввода элементов массива
        {
          printf("%c[%d][%d]=",c,i,j);
          scanf("%d",&p[i**n+j]);      // ввод элементов массива
        }
    return(p);                         // возврат массива
}
void vivod2m(int *p,int m,int n,char c)  // функция вывода элементов
                                         // массива
{
```

```
 int i,                                  // параметр цикла
    j;                                   // параметр цикла
 printf("  %c\n",c);
 for(i=0; i<m; i++)                      // внешний цикл вывода элементов массива
  {
    for(j=0;j<n;j++)                     // внутренний цикл вывода элементов массива
       printf("%6d ",p[i*n+j]); // вывод элементов массива
    printf("\n");
 }
}
int maxel(int *p, int m, int n)    // функция, вычисляющая макс. элемент
                                   // массива
{
 int max=p[0],                     // присваивание значения p-нулевого
                                   // переменной max
          i,                       // параметр цикла
          j;                       // параметр цикла
 for(i=0; i<m; i++)                // внешний цикл поиска макс. элемента
                                   // массива
   for(j=0;j<n;j++)                // внутренний цикл поиска макс. Элемента
                                   // массива
       if(p[i*n+j]>max)            // условие нахождения макс. элемента
                                   // массива
         max=p[i*n+j];             // присваивание переменной max макс.
                                   // элемента массива
 return(max);                      // возврат макс. элемента массива
}
```

4.4 Тесты

1) Исходные данные:

Массив X

-1 1

-1 -1

Массив Y

3 3

2 0

Массив Z

-3 -7

-8 -9

Результат:

Максимальный элемент массива X равен 1

Максимальный элемент массива Y равен 3

Максимальный элемент массива Z равен -3

Корни уравнения: 0.791 и -3.791

2) Исходные данные:

Массив X

0 -5 -9

-7 -8 -1

Массив Y

5 5

5 5

1 0

Массив Z

7 8

8 8

Результат:

Максимальный элемент массива X равен 0

Максимальный элемент массива Y равен 5

Максимальный элемент массива Z равен 8

Квадратное уравнение не сформировано: a=0

3) Исходные данные:

Массив X

2 2 2

0 -1 0

Массив Y

1 -3

-5 0

Массив Z

9 0

0 1

Результат:

Максимальный элемент массива X равен 2

Максимальный элемент массива Y равен 1

Максимальный элемент массива Z равен 9

Нет корней: дискриминант меньше 0

4) Исходные данные:

Массив X

-1 1

0 0

Массив Y

-8 -10

-10 -11

Массив Z

16 0

1 7

Результат:

Максимальный элемент массива X равен 1

Максимальный элемент массива Y равен -8

Максимальный элемент массива Z равен 16

Корни уравнения: 4.000 и 4.000

5) Исходные данные:

Массив X

1 -9

-10 0

Массив Y

0 -5

-5 -7

Массив Z

1 0

0 0

Результат:

Максимальный элемент массива X равен 1

Максимальный элемент массива Y равен 0

Максимальный элемент массива Z равен 1

Нет корней: дискриминант меньше 0

5. Передача имени функций в качестве параметров

5.1 Теоретические сведения

Функцию можно вызвать через указатель на нее. Для этого объявляется указатель соответствующего типа и ему с помощью операции взятия адреса присваивается адрес функции:

```
void f(int a) {/* … */}     // определение функции
void (*pf)(int);             // указатель на функцию
 …
 pf = &f;                    // указателю присваивается адрес функции
 pf(10);                     // функция f вызывается через указатель pf
                             // (можно написать (*pf)(10) )
```

Для того чтобы сделать программу легко читаемой, при описании указателей на функции используют переименование типов (typedef). Можно объявлять массивы указателей на функции (это может быть полезно, например, при реализации меню):

```
// Описание типа PF как указателя
// на функцию с одним параметром типа int:
typedef void (*PF)(int);
// Описание и инициализация массива указателей:
PF menu[] = {&new, &open, &save};
menu[l](10);        // вызов функции open
```

Здесь new, open и save — имена функций, которые должны быть объявлены ранее.

Указатели на функции передаются в подпрограмму таким же образом, как и параметры других типов:

```
#include<stdio.h>
#include <iostream.h>
typedef void (*PF)(int);
void f1(PF pf)   // функция fl получает в качестве параметра указатель
                 // типа PF
{
   pf(5);          // вызов функции, переданной через указатель
}
void f(int i ){cout «1;}
int main()
{
   fl(f);
   return (0);
}
```

Тип указателя и тип функции, которая вызывается посредством этого указателя,
должны совпадать в точности.

5.2 Метод центральных прямоугольников

Многие инженерные задачи, задачи физики, геометрии и многих других областей человеческой деятельности приводят к необходимости вычислять определенный интеграл вида $\int_a^b f(x)dx$ где f(x) -данная функция, непрерывная на отрезке [a; b]. Если функция f(x) задана формулой и мы умеем найти неопределенный интеграл F(x), то определенный интеграл вычисляется по формуле Ньютона-Лейбница: $\int_a^b f(x)dx=F(b)-F(a)$

Если же неопределенный интеграл данной функции мы найти не умеем, или по какой-либо причине не хотим воспользоваться формулой Ньютона-Лейбница или если функция f(x) задана графически или таблицей, то для вычисления определенного интеграла применяют приближенные формулы. Для приближенного вычисления интеграла можно использовать метод прямоугольников (правых, левых, средних). При вычислении интеграла следует помнить, каков геометрический смысл определенного интеграла. Если f(x)>=0 на отрезке [a; b], то $\int_a^b f(x)dx$ численно равен площади фигуры, ограниченной графиком функции y=f(x), отрезком оси абсцисс, прямой x=a и прямой x=b (рисунок 11) Таким образом, вычисление интеграла равносильно вычислению площади криволинейной трапеции.

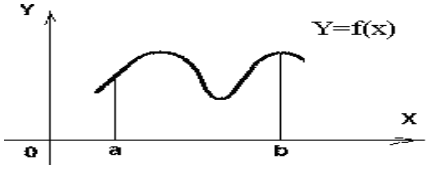
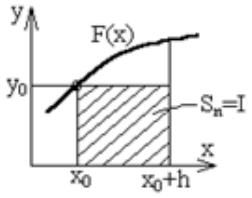


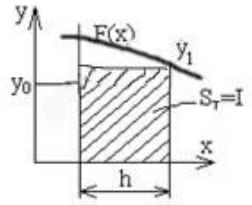
Рисунок 11

Разделим отрезок [a; b] на n равных частей, т.е. на n элементарных отрезков. Длина каждого элементарного отрезка $h=\frac{b-a}{n}$.
Точки деления будут: $x_0=a$; $x_1=a+h$; $x_2=a+2*h$, ... , $x_{n-1}=a+(n-1)*h$; $x_n=b$.

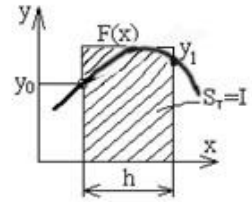
Числа y_0, y_1, y_2, ... , y_n являются ординатами точек графика функции, соответствующих абсциссам x_0, x_1, x_2, ... , x_n (рисунок 12).
Строим прямоугольники. Это можно делать несколькими способами:
Левые прямоуголики (слева на право)



Правые прямоугоники (построение справа на лево)



Средние прямоугольники (посредине)



Площадь криволинейной трапеции приближенно заменяется площадью многоугольника, составленного из n прямоугольников. Таким образом, вычисление определенного интеграла сводится к нахождению суммы n элементарных прямоугольников.

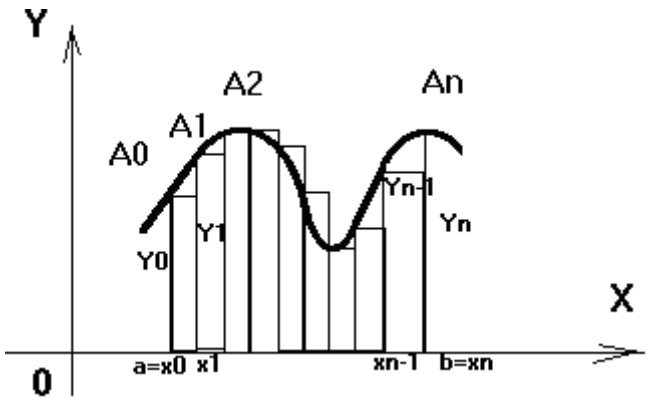


Рисунок 12

h=(b-a)/n –ширина прямоугольников

Формула левых прямоугольников:

$$S \approx \int_a^b f(x)dx \approx y_0 \cdot h + y_1 \cdot h + y_2 \cdot h + y_3 \cdot h + ... + y_{n-1} \cdot h \approx h \cdot (y_0 + y_1 + y_2 + y_3 + ... + y_{n-1}) \quad (1.3)$$

Формула правых прямоугольников:

$$S \approx \int_a^b f(x)dx \approx y_1 \cdot h + y_2 \cdot h + y_3 \cdot h + y_4 \cdot h + ... + y_n \cdot h \approx h \cdot (y_1 + y_2 + y_3 + y_4 + ... + y_n) \quad (1.4)$$

Формула средних прямоугольников.

$S_{средих} = (S_{правых} + S_{левых}) /2$

$$S \approx h \sum_{i=0}^{n-1} y(x_i + \frac{h}{2}) \quad (1.5)$$

5.3 Графическая схема алгоритма

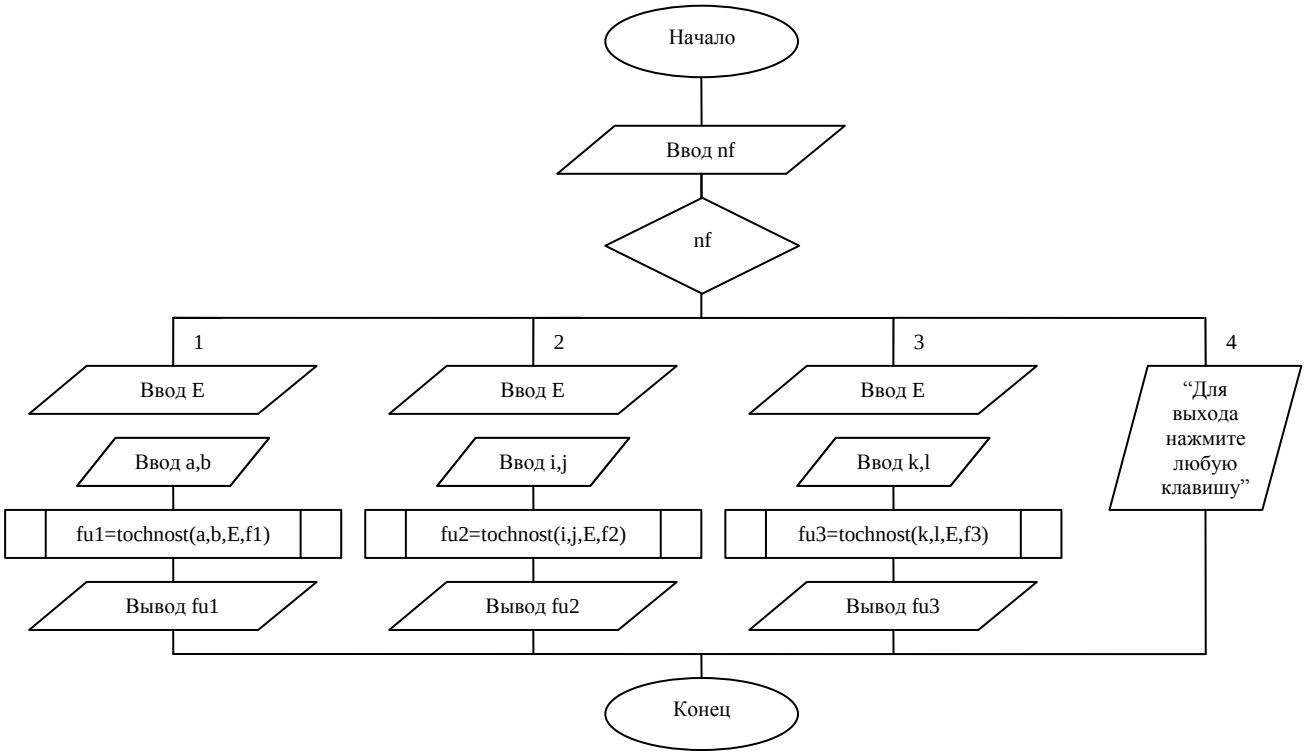


Рисунок 13 - Графическая схема основного алгоритма

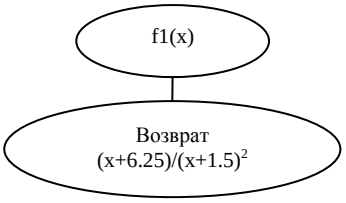


Рисунок 14 - Графическая схема алгоритма первой

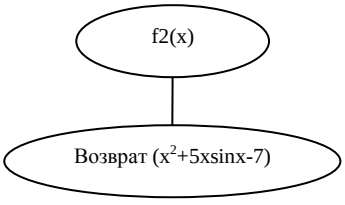


Рисунок 15 - Графическая схема алгоритма второй

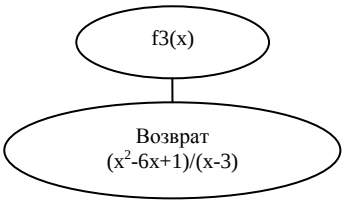


Рисунок 16 - Графическая схема алгоритма третьей функции

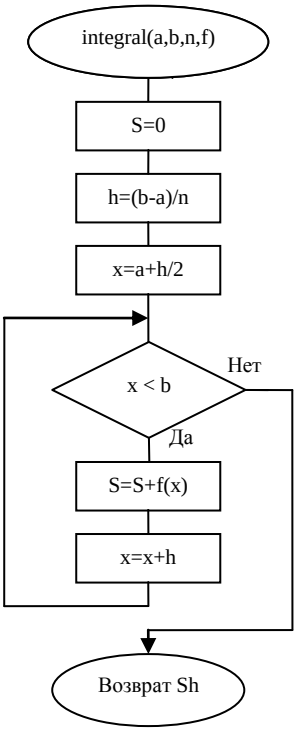


Рисунок 17 - Графическая схема алгоритма вычисления

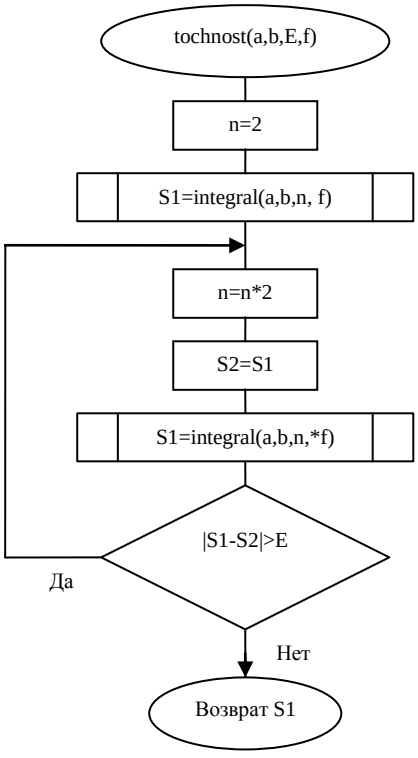


Рисунок 18 - Графическая схема алгоритма вычисления интеграла с точностью E

5.4 Текст программы

```
#include<math.h>              // директивы
#include<stdio.h>             // подключения
#include<conio.h>             // файлов
typedef float(*func)(float);  // переименование типов
float f1(float x);            // прототип первой функции
float f2(float x);            // прототип второй функции
float f3(float x);            // прототип третьей функции
float integral(float a,float b,long n,func f);   // прототип функции
                                                 // вычисления интеграла
float tochnost(float a,float b,float E,func f); // прототип функции
                     // вычисления интеграла с точностью E
void main()          // главная функция, не возвращающая значение
{
 clrscr();
 float S1,   // значение интеграла
       E,    // точность вычисления
       a,    // левый предел первой функции
       b,    // правый предел первой функции
       i,    // левый предел второй функции
       j,    // правый предел второй функции
       k,    // левый предел третьей функции
       l,    // правый предел третьей функции
     fu1,    // интеграл первой функции
     fu2,    // интеграл второй функции
     fu3;    // интеграл третьей функции
 int nf;   // номер функции, интеграл которой необходимо вычислить
 puts("Введите номер функции, интеграл которой необходимо
вычислить");
 printf("\n");
 puts("       x+6.25");
 puts("1. y= ──────────────");
 puts("     (x+1.5)^2 ");
 printf("\n");
 puts("2. y= x^2+5xsinx-7");
 printf("\n");
 puts("      x^2-6x+1");
 puts("3. y= ──────────────");
 puts("       x-3    ");
 puts("4. Выход ");
```

```
scanf("%d", &nf);
switch (nf)                          // оператор выбора
 {
  case 1:
  {
    puts("Введите точность");
    scanf("%f",&E);                  // ввод точности вычисления
    puts("Введите интервалы для первой функции");
    scanf("%f %f",&a,&b);            // ввод интервалов для первой функции
    fu1=tochnost(a,b,E,*f1);
    printf("Интеграл f1 равен: %8.6f\n",fu1);   // вывод интеграла первой
                                                 // функции
  }
  break;                                         // оператор выхода из switch
  case 2:
  {
    puts("Введите точность");
    scanf("%f",&E);                              // ввод точности вычисления
    puts("Введите интервалы для второй функции");
    scanf("%f %f",&i,&j);              // ввод интервалов для второй функции
    fu2=tochnost(i,j,E,*f2);
    printf("Интеграл f2 равен: %8.6f\n",fu2);  // вывод интеграла второй
                                               // функции
  }
  break;                                       // оператор выхода из switch
  case 3:
  {
    puts("Введите точность");
    scanf("%f",&E);                    // ввод точности вычисления
    puts("Введите интервалы для третьей функции");
    scanf("%f %f",&k,&l);              // ввод интервалов для третьей функции
    fu3=tochnost(k,l,E,*f3);
    printf("Интеграл f3 равен: %8.6f\n",fu3);  // вывод интеграла третьей
                                               // функции
  }
  break;                                       // оператор выхода из switch
  case 4:   printf("Для  выхода нажмите любую клавишу\n");
  break;                                       // оператор выхода из switch
 }
 fflush(stdin);
 getch();
}
```

```
float tochnost(float a,float b,float E,func f)       // функция вычисления
                                                     // интеграла с точностью E
{
 float S1,           // интеграл функции при кол-ве разбиений равном n*2
       S2;           // интеграл функции при кол-ве разбиений равном n
 long n=2;           // n - количество разбиений
 S1=integral(a,b,n,*f);       // вычисление интеграла функции при кол-ве
                              // разбиений n
 do                           // цикл вычисления интеграла при точности
                              // вычисления E
 {
  n=n*2;                    // увеличивание кол-ва разбиений в 2 раза
  S2=S1;                    // присваивание интеграла функции переменной S2
  S1=integral(a,b,n,*f);    // вычисление интеграла функции при
                             // количестве разбиений n*2
 }
 while(fabs(S1-S2)>E);      // условие повторения цикла
 return(S1);                // возврат интеграла
}
float f1(float x)           // первая функция
 {
   return((x+6.25)/pow((x+1.5),2));    // возврат первой функции
 }
float f2(float x)           // вторая функция
{
 return(x*x+5*x*sin(x)-7);             // возврат второй функции
}
float f3(float x)           // третья функция
{
 return((x*x-6*x+1)/(x-3));            // возврат третьей функции
}
float integral(float a,float b,long n ,func f)
{
 float S=0,                            // высота n количества кусков
       h,                              // шаг
       x;                              // ширина n количества кусков
 h=(b-a)/n;                            // вычисление шага
 x=a+h/2;                   // вычисление ширины n количества кусков
 while(x<b)
 {
  S=S+f(x);
  x+=h;
```

```
 }
 return(S*h);                          // возврат интеграла
}
```

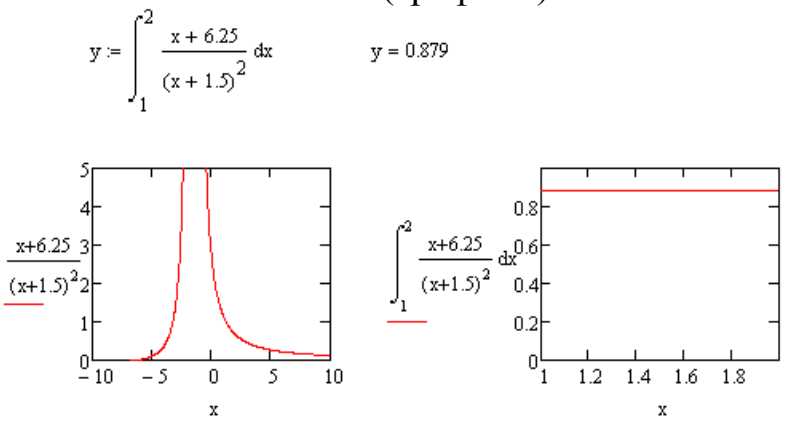
5.5 Тесты

1) Исходные данные:
   Номер функции(nf): 1
   Точность равна 0.1
   Интервалы: от 1 до 2

Результат:

Интеграл f1 равен: 0.878124

Тест 1 (графики)

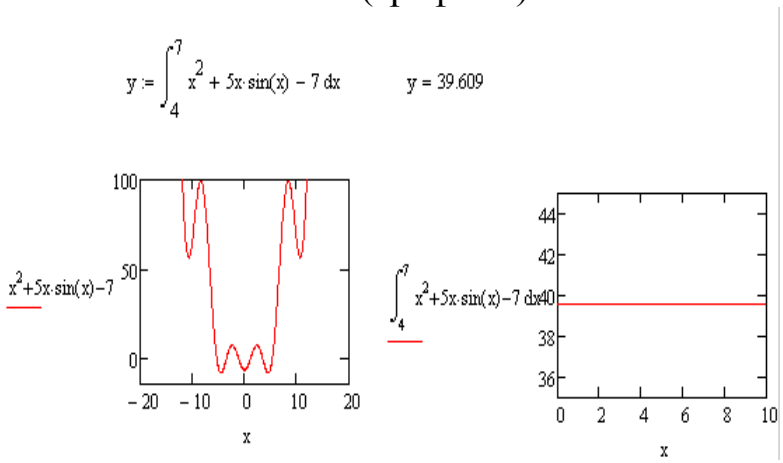


2) Исходные данные:
   Номер функции(nf): 2
   Точность равна 0.00001
   Интервалы: от 4 до 7

Результат:

Интеграл f2 равен: 39.609520

Тест 2(графики)

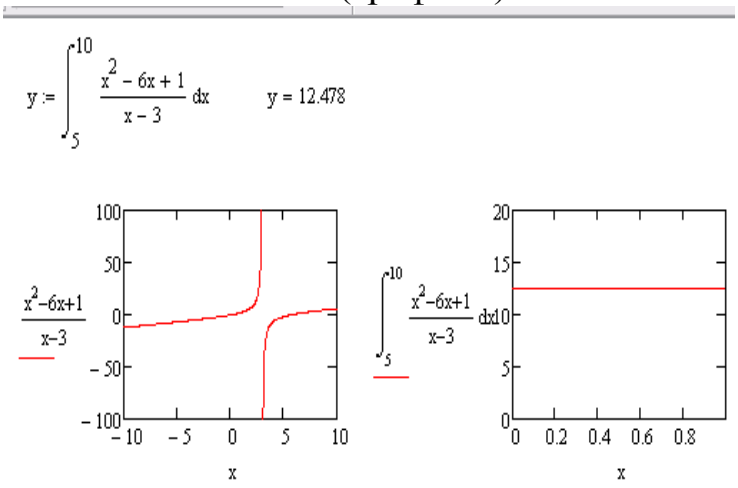


3) Исходные данные:
Номер функции(nf): 3
Точность равна 0.001
Интервалы: от 5 до 10

Результат:

Интеграл f3 равен: 12.478012

Тест 3(графики)



4) Исходные данные:
nf = 4

Результат:
Выход из программы

Заключение

Работа выполнена на языке высокого уровня C++, прочно вошедшем в мир программирования . В результате её написания были разработаны и реализованы программы с использованием подпрограмм, в которые передавались :

а) переменные
б) одномерный массив
в) двумерный массив
г) имя другой функции (в качестве параметра)

В первой задаче переменные в функцию передавались по значению и по ссылке. Во второй , при использовании в качестве параметра одномерного массива в функцию передавался его первый элемент, иными словами, массив передавался по адресу. В задаче под номером три при передаче в подпрограмму двумерного массива осуществлялась замена его одномерным и имитировался доступ к двумерному массиву внутри функции. В четвёртой задаче подпрограмма вызывалась через указатель на неё.

Разделение программ на более простые и обозримые части с помощью функций значительно облегчило их написание, позволило избежать избыточности кода, а также упростило их структуру.

При разработке программ производились вычисления и обработка данных.

При написании курсовой работы был приобретен опыт работы с языком программирования, изучен синтаксис данного языка, основные конструкции, его семантика.

Список использованных источников

1. Павловская, Щупак. C/C++. Структурное программирование: Практикум/СПб.: Питер, 2003.- с.132.

2. И.Ф.Астахова, С.В.Власов. Язык С++ : Учебное пособие.- Мн.: Новое издание 2003. - с.6.

3.Стефан Р.Дэвис . С++ для "чайников", 4-е издание. : Пер. с англ. : — М. : Издательский дом "Вильяме", 2003.- с.68

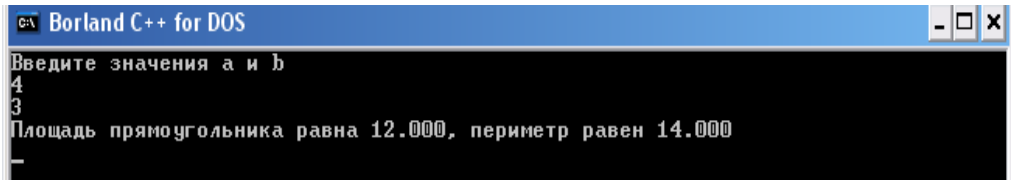
4. Т.А.Павловская С/С++. Программирование на языке высокого уровня . — СПб.: Питер, 2003.- с.73.

Приложение

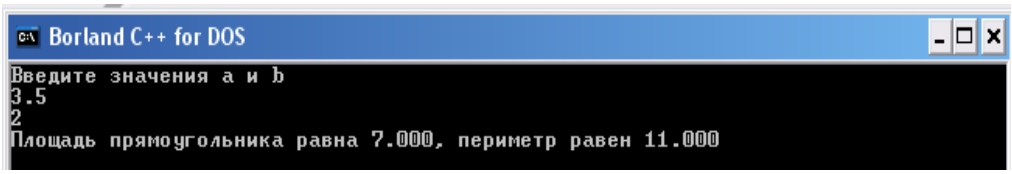
Результаты выполнения программ

1. Задача 1

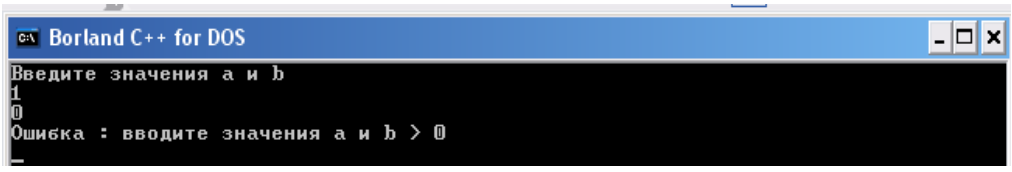
1)



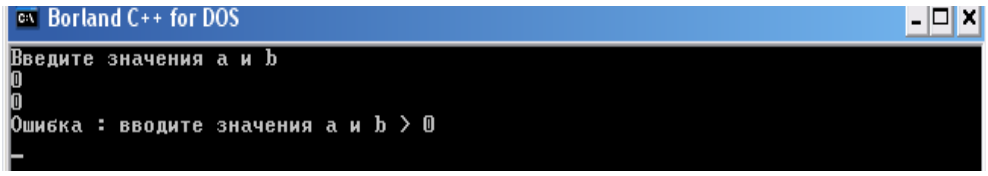
2)



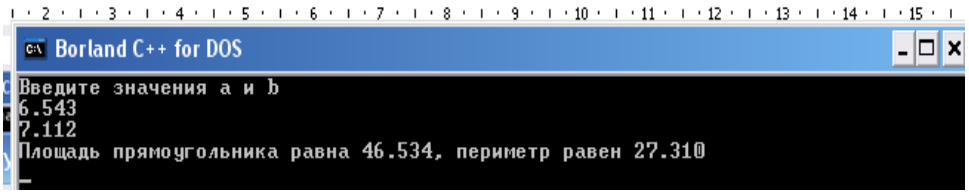
3)



4)

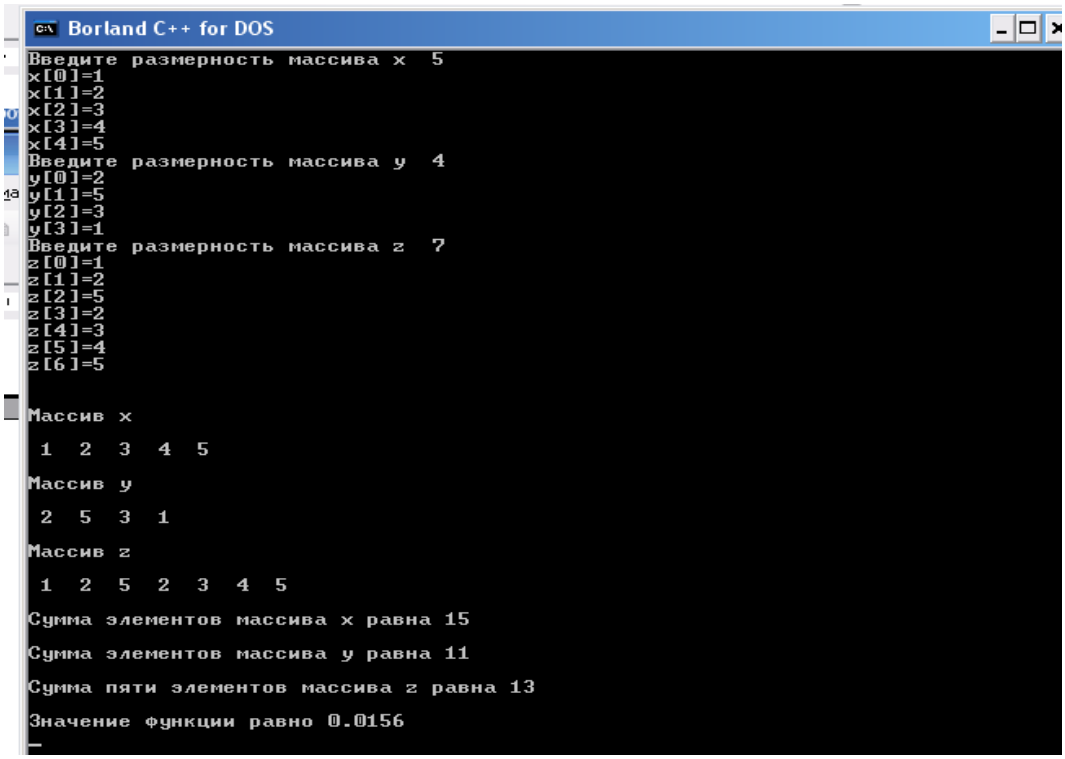


5)

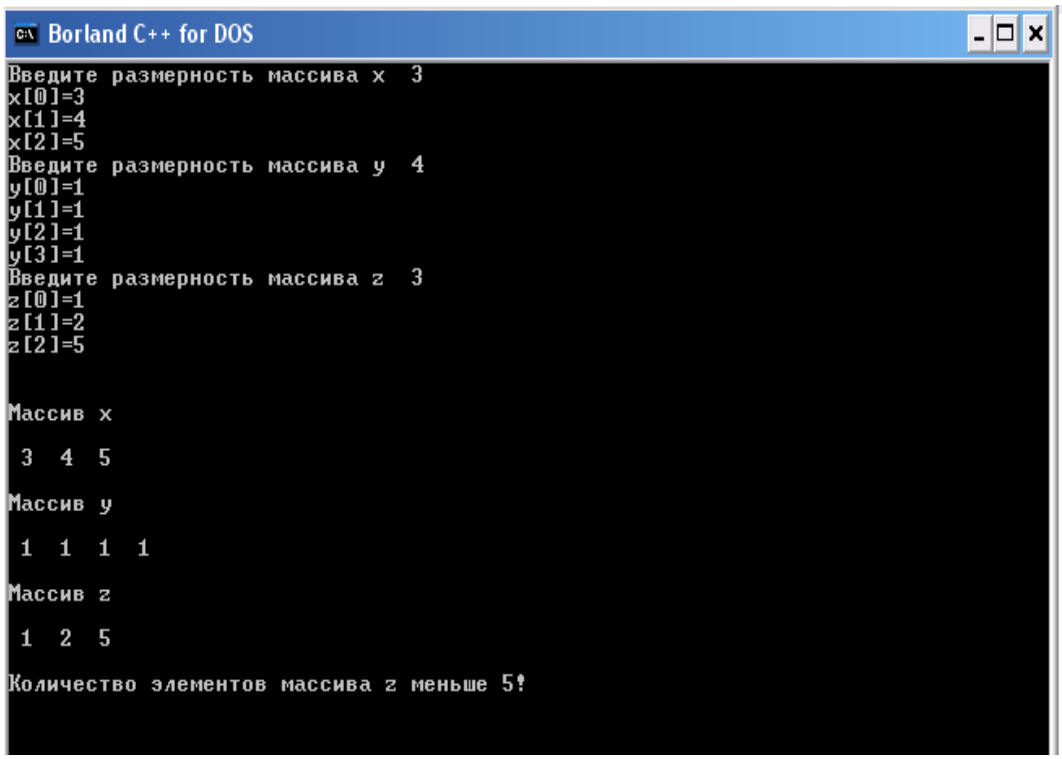


2. Задача 2

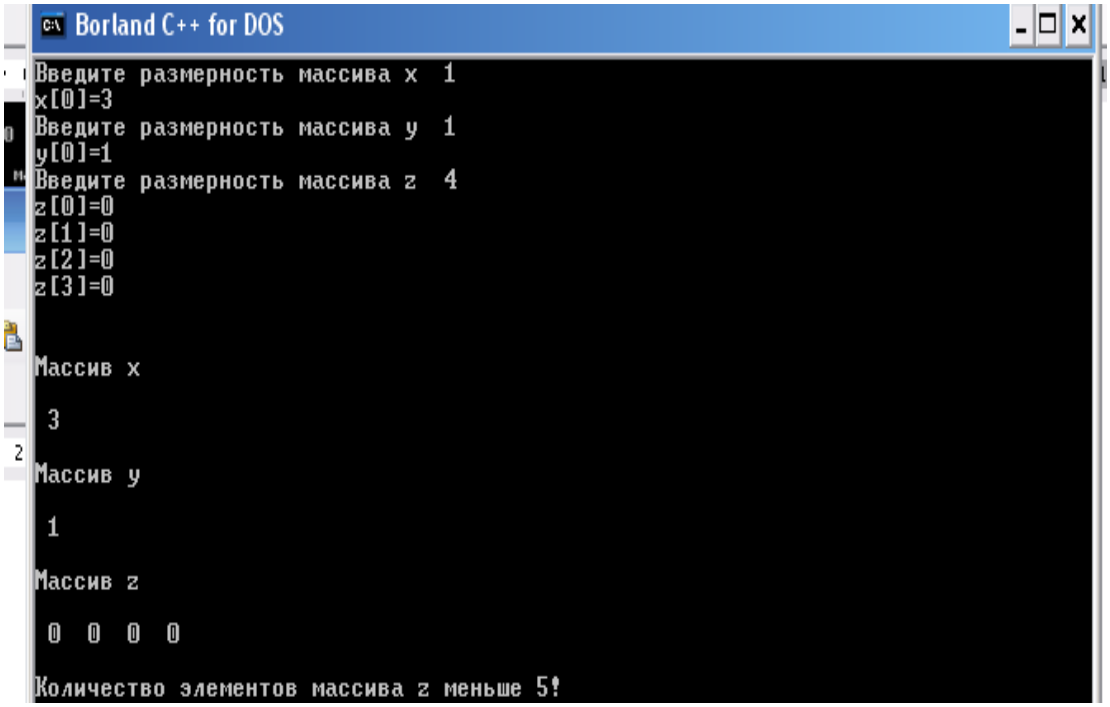
1)


Borland C++ for DOS
Введите размерность массива x 5
x[0]=1
x[1]=2
x[2]=3
x[3]=4
x[4]=5
Введите размерность массива y 4
y[0]=2
y[1]=5
y[2]=3
y[3]=1
Введите размерность массива z 7
z[0]=1
z[1]=2
z[2]=5
z[3]=2
z[4]=3
z[5]=4
z[6]=5
Массив x
1 2 3 4 5
Массив y
2 5 3 1
Массив z
1 2 5 2 3 4 5
Сумма элементов массива x равна 15
Сумма элементов массива y равна 11
Сумма пяти элементов массива z равна 13
Значение функции равно 0.0156

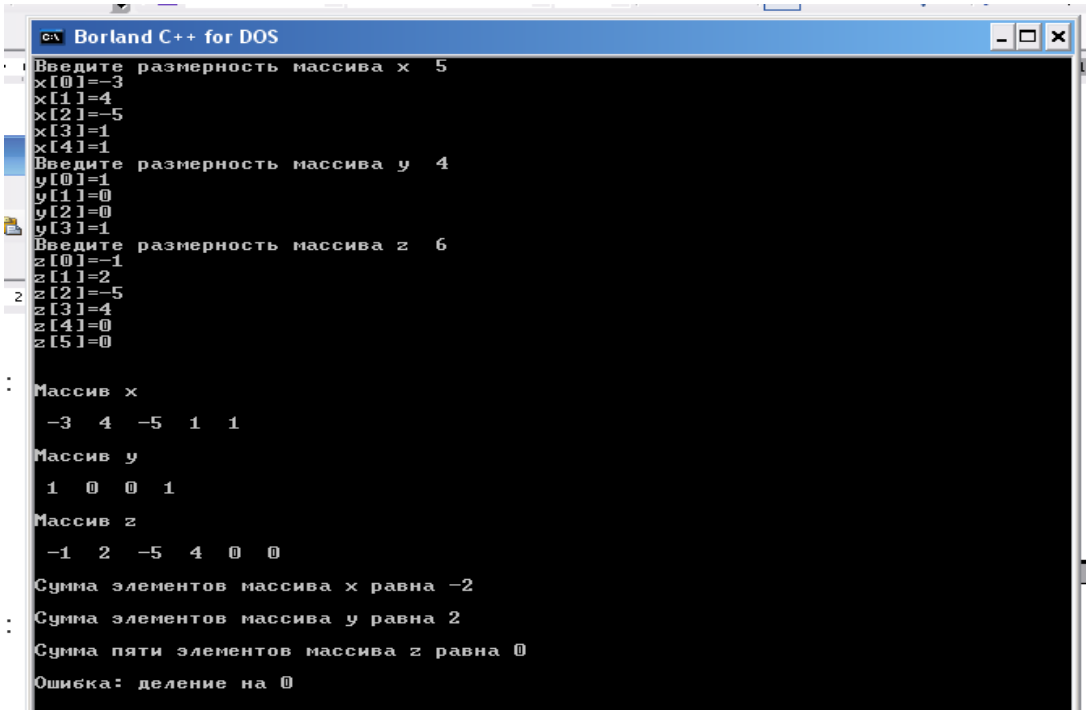
2)


Borland C++ for DOS
Введите размерность массива x 3
x[0]=3
x[1]=4
x[2]=5
Введите размерность массива y 4
y[0]=1
y[1]=1
y[2]=1
y[3]=1
Введите размерность массива z 3
z[0]=1
z[1]=2
z[2]=5
Массив x
3 4 5
Массив y
1 1 1 1
Массив z
1 2 5
Количество элементов массива z меньше 5!

3)

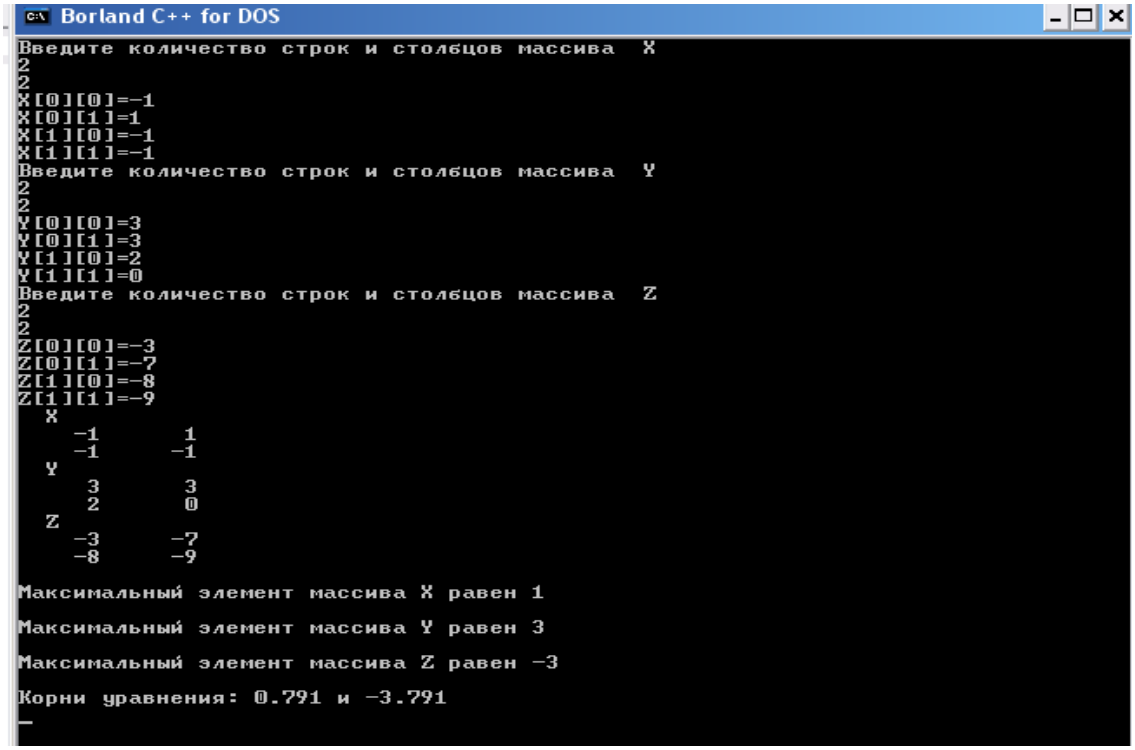

Borland C++ for DOS
Введите размерность массива x 1
x[0]=3
Введите размерность массива y 1
y[0]=1
Введите размерность массива z 4
z[0]=0
z[1]=0
z[2]=0
z[3]=0
Массив x
3
Массив y
1
Массив z
0 0 0 0
Количество элементов массива z меньше 5!

4)

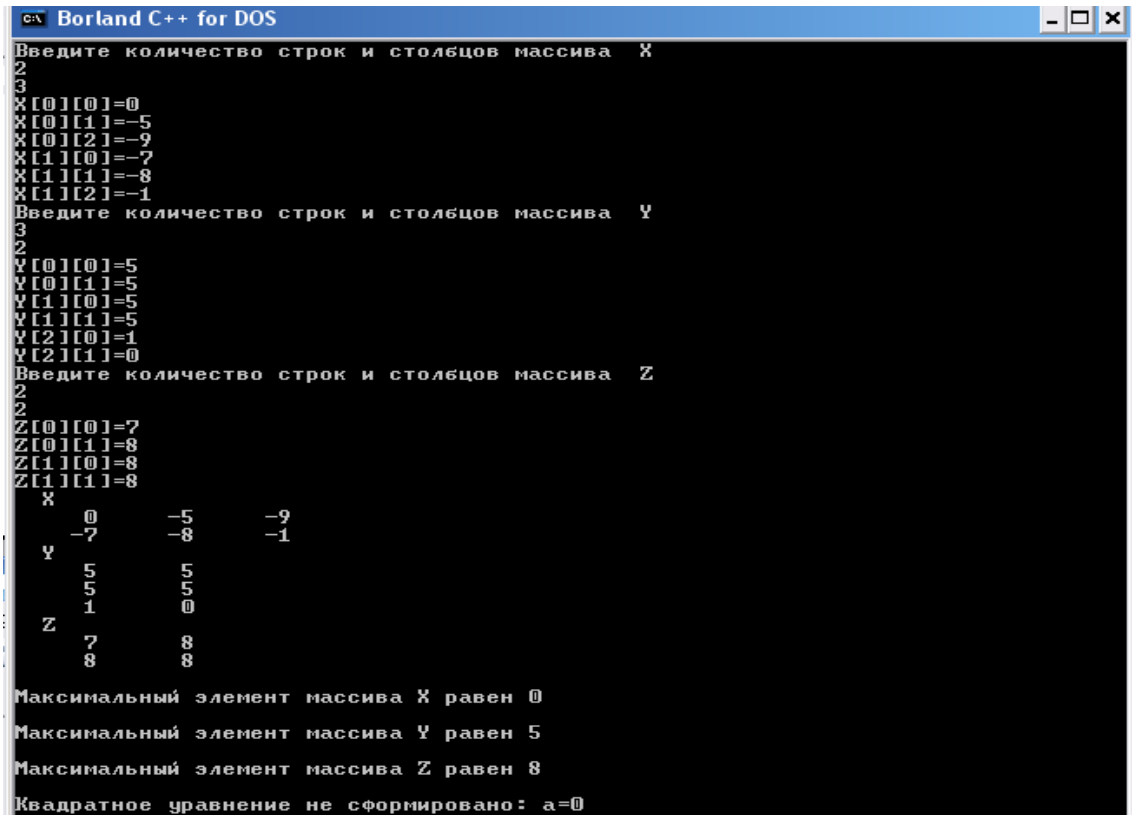

Borland C++ for DOS
Введите размерность массива x 5
x[0]=-3
x[1]=4
x[2]=-5
x[3]=1
x[4]=1
Введите размерность массива y 4
y[0]=1
y[1]=0
y[2]=0
y[3]=1
Введите размерность массива z 6
z[0]=-1
z[1]=2
z[2]=-5
z[3]=4
z[4]=0
z[5]=0
Массив x
-3 4 -5 1 1
Массив y
1 0 0 1
Массив z
-1 2 -5 4 0 0
Сумма элементов массива x равна -2
Сумма элементов массива y равна 2
Сумма пяти элементов массива z равна 0
Ошибка: деление на 0

3. Задача 3

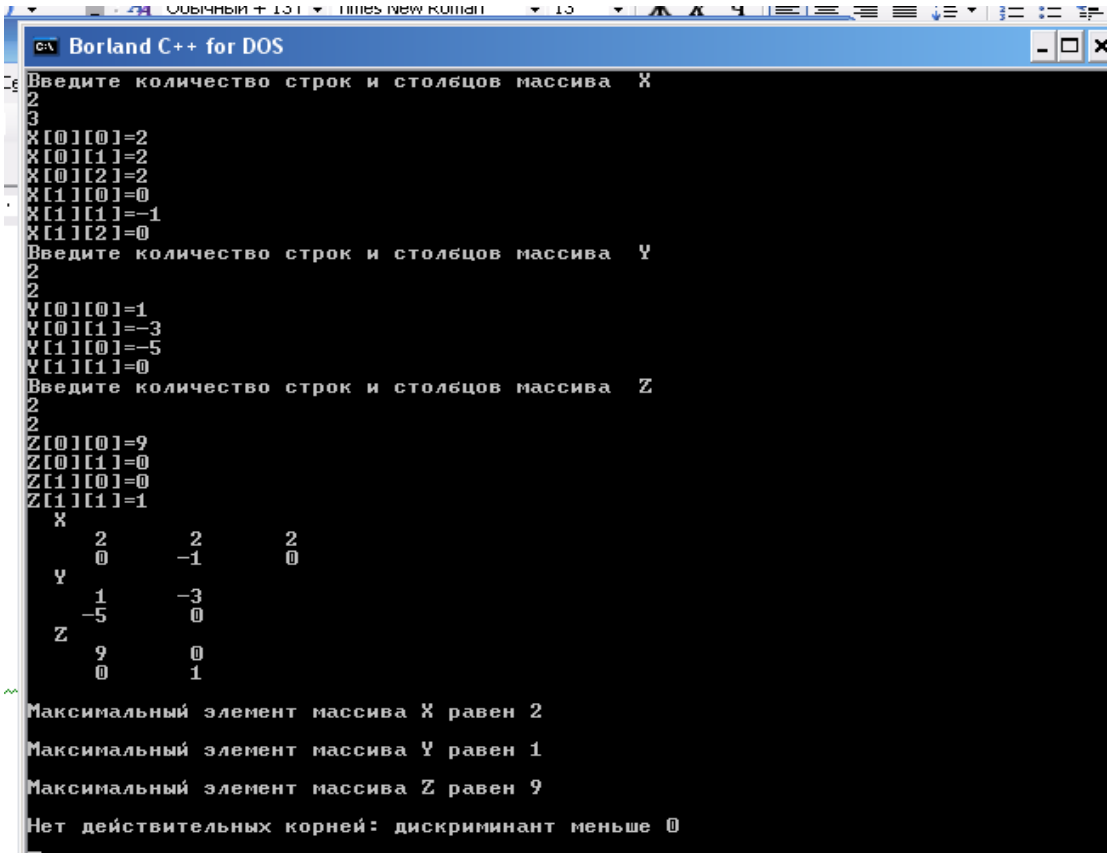
1)


Borland C++ for DOS
Введите количество строк и столбцов массива X
2
2
X[0][0]=-1
X[0][1]=1
X[1][0]=-1
X[1][1]=-1
Введите количество строк и столбцов массива Y
2
2
Y[0][0]=3
Y[0][1]=3
Y[1][0]=2
Y[1][1]=0
Введите количество строк и столбцов массива Z
2
2
Z[0][0]=-3
Z[0][1]=-7
Z[1][0]=-8
Z[1][1]=-9
X
-1 1
-1 -1
Y
3 3
2 0
Z
-3 -7
-8 -9
Максимальный элемент массива X равен 1
Максимальный элемент массива Y равен 3
Максимальный элемент массива Z равен -3
Корни уравнения: 0.791 и -3.791

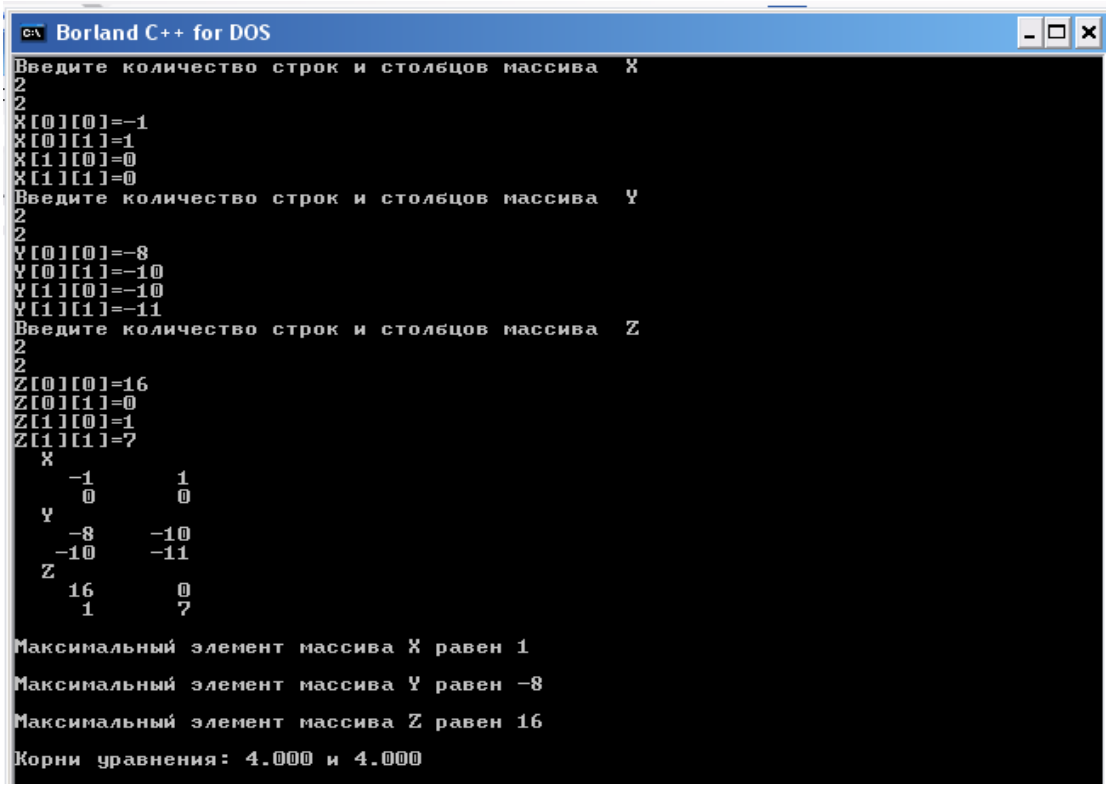
2)


Borland C++ for DOS
Введите количество строк и столбцов массива X
2
3
X[0][0]=0
X[0][1]=-5
X[0][2]=-9
X[1][0]=-7
X[1][1]=-8
X[1][2]=-1
Введите количество строк и столбцов массива Y
3
2
Y[0][0]=5
Y[0][1]=5
Y[1][0]=5
Y[1][1]=5
Y[2][0]=1
Y[2][1]=0
Введите количество строк и столбцов массива Z
2
2
Z[0][0]=7
Z[0][1]=8
Z[1][0]=8
Z[1][1]=8
X
0 -5 -9
-7 -8 -1
Y
5 5
5 5
1 0
Z
7 8
8 8
Максимальный элемент массива X равен 0
Максимальный элемент массива Y равен 5
Максимальный элемент массива Z равен 8
Квадратное уравнение не сформировано: a=0

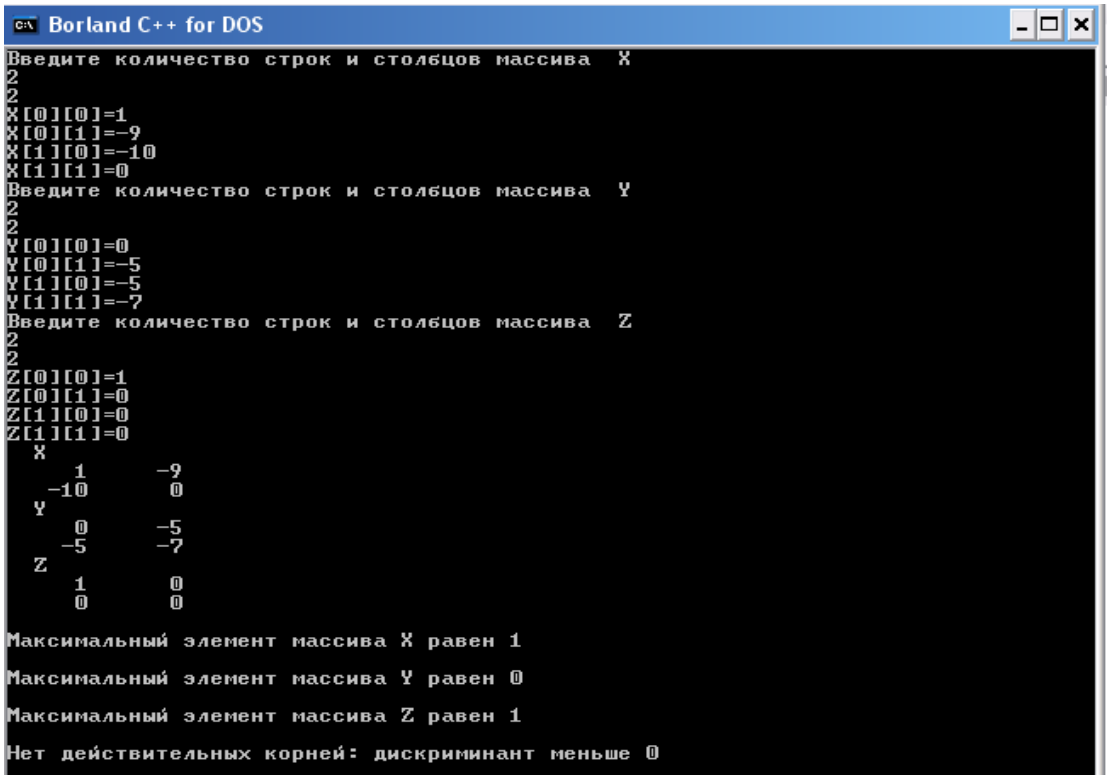
3)


Borland C++ for DOS
Введите количество строк и столбцов массива X
2
3
X[0][0]=2
X[0][1]=2
X[0][2]=2
X[1][0]=0
X[1][1]=-1
X[1][2]=0
Введите количество строк и столбцов массива Y
2
2
Y[0][0]=1
Y[0][1]=-3
Y[1][0]=-5
Y[1][1]=0
Введите количество строк и столбцов массива Z
2
2
Z[0][0]=9
Z[0][1]=0
Z[1][0]=0
Z[1][1]=1
X
2 2 2
0 -1 0
Y
1 -3
-5 0
Z
9 0
0 1
Максимальный элемент массива X равен 2
Максимальный элемент массива Y равен 1
Максимальный элемент массива Z равен 9
Нет действительных корней: дискриминант меньше 0

4)

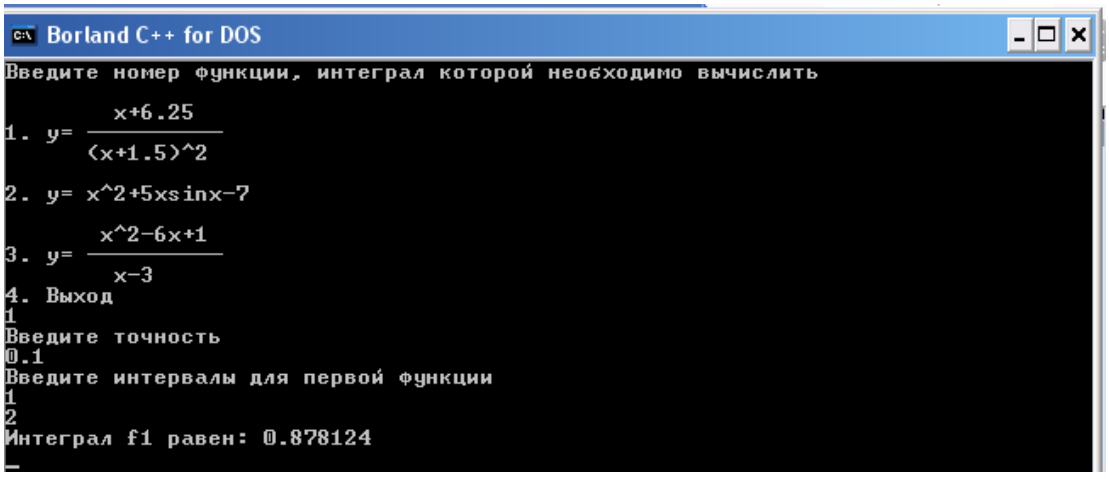

Borland C++ for DOS
Введите количество строк и столбцов массива X
2
2
X[0][0]=-1
X[0][1]=1
X[1][0]=0
X[1][1]=0
Введите количество строк и столбцов массива Y
2
2
Y[0][0]=-8
Y[0][1]=-10
Y[1][0]=-10
Y[1][1]=-11
Введите количество строк и столбцов массива Z
2
2
Z[0][0]=16
Z[0][1]=0
Z[1][0]=1
Z[1][1]=7
X
-1 1
0 0
Y
-8 -10
-10 -11
Z
16 0
1 7
Максимальный элемент массива X равен 1
Максимальный элемент массива Y равен -8
Максимальный элемент массива Z равен 16
Корни уравнения: 4.000 и 4.000

5)

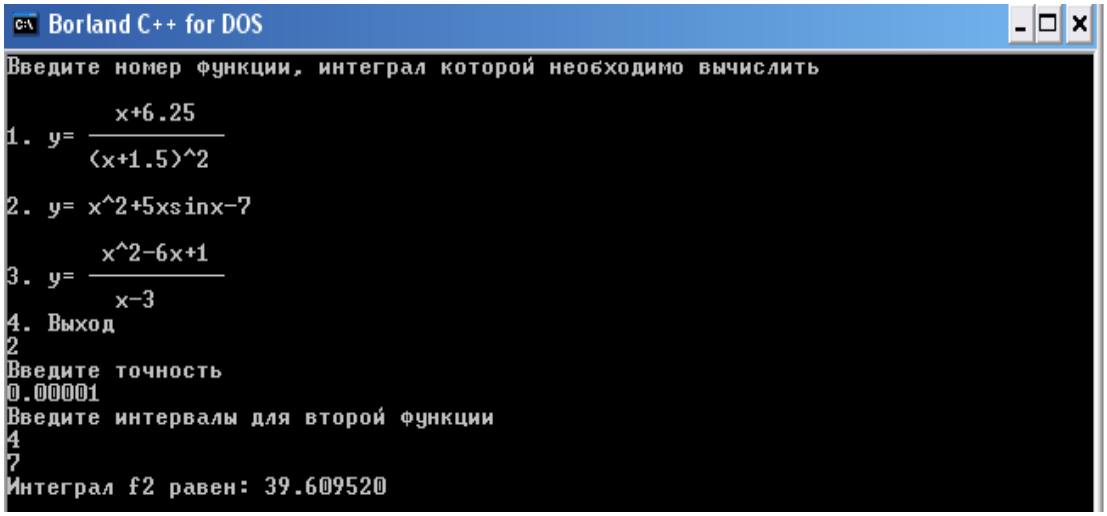

Borland C++ for DOS
Введите количество строк и столбцов массива X
2
2
X[0][0]=1
X[0][1]=-9
X[1][0]=-10
X[1][1]=0
Введите количество строк и столбцов массива Y
2
2
Y[0][0]=0
Y[0][1]=-5
Y[1][0]=-5
Y[1][1]=-7
Введите количество строк и столбцов массива Z
2
2
Z[0][0]=1
Z[0][1]=0
Z[1][0]=0
Z[1][1]=0
X
1 -9
-10 0
Y
0 -5
-5 -7
Z
1 0
0 0
Максимальный элемент массива X равен 1
Максимальный элемент массива Y равен 0
Максимальный элемент массива Z равен 1
Нет действительных корней: дискриминант меньше 0

4. Задача 4

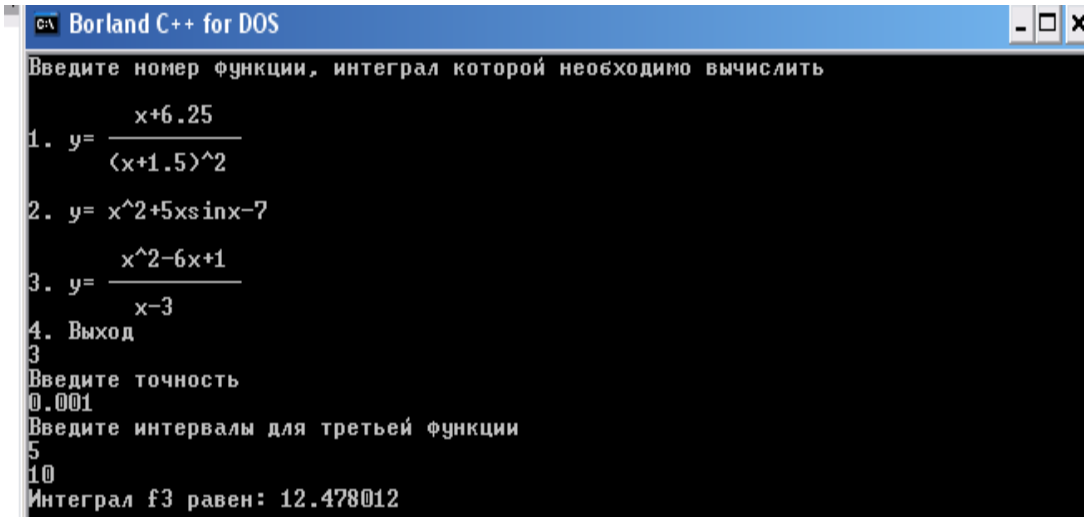
1)


Borland C++ for DOS
Введите номер функции, интеграл которой необходимо вычислить
1. y= (x+6.25)/(x+1.5)^2
2. y= x^2+5xsinx-7
3. y= (x^2-6x+1)/(x-3)
4. Выход
1
Введите точность
0.1
Введите интервалы для первой функции
1
2
Интеграл f1 равен: 0.878124

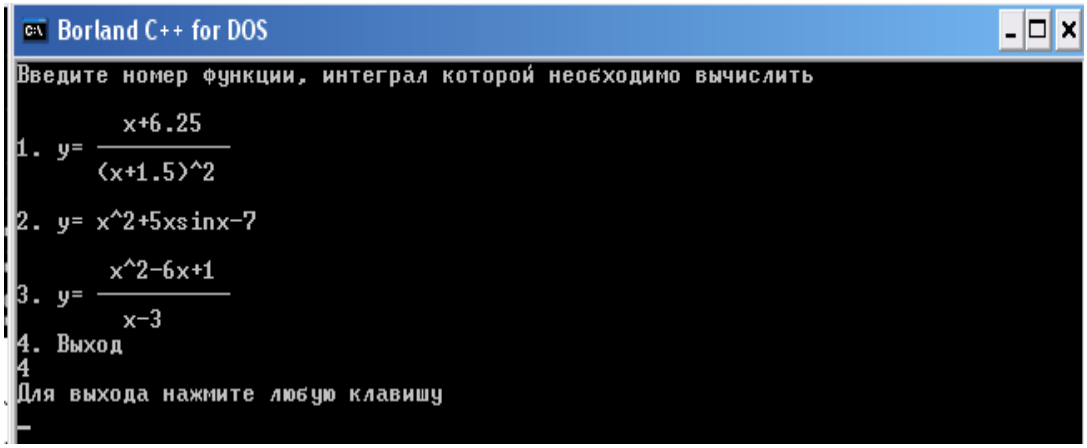
2)


Borland C++ for DOS
Введите номер функции, интеграл которой необходимо вычислить
1. y= (x+6.25)/(x+1.5)^2
2. y= x^2+5xsinx-7
3. y= (x^2-6x+1)/(x-3)
4. Выход
2
Введите точность
0.00001
Введите интервалы для второй функции
4
7
Интеграл f2 равен: 39.609520

3)


Borland C++ for DOS
Введите номер функции, интеграл которой необходимо вычислить
1. y= (x+6.25)/(x+1.5)^2
2. y= x^2+5xsinx-7
3. y= (x^2-6x+1)/(x-3)
4. Выход
3
Введите точность
0.001
Введите интервалы для третьей функции
5
10
Интеграл f3 равен: 12.478012

4)


Borland C++ for DOS
Введите номер функции, интеграл которой необходимо вычислить
1. y= (x+6.25)/(x+1.5)^2
2. y= x^2+5xsinx-7
3. y= (x^2-6x+1)/(x-3)
4. Выход
4
Для выхода нажмите любую клавишу