Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №4**

**«РАБОТА В ГРАФИЧЕСКОМ РЕЖИМЕ»**

Выполнил: студент учебной группы

ИСПк-204-52-00

Федяев Никита Юрьевич

Преподаватель:

Сергеева Елизавета Григорьевна

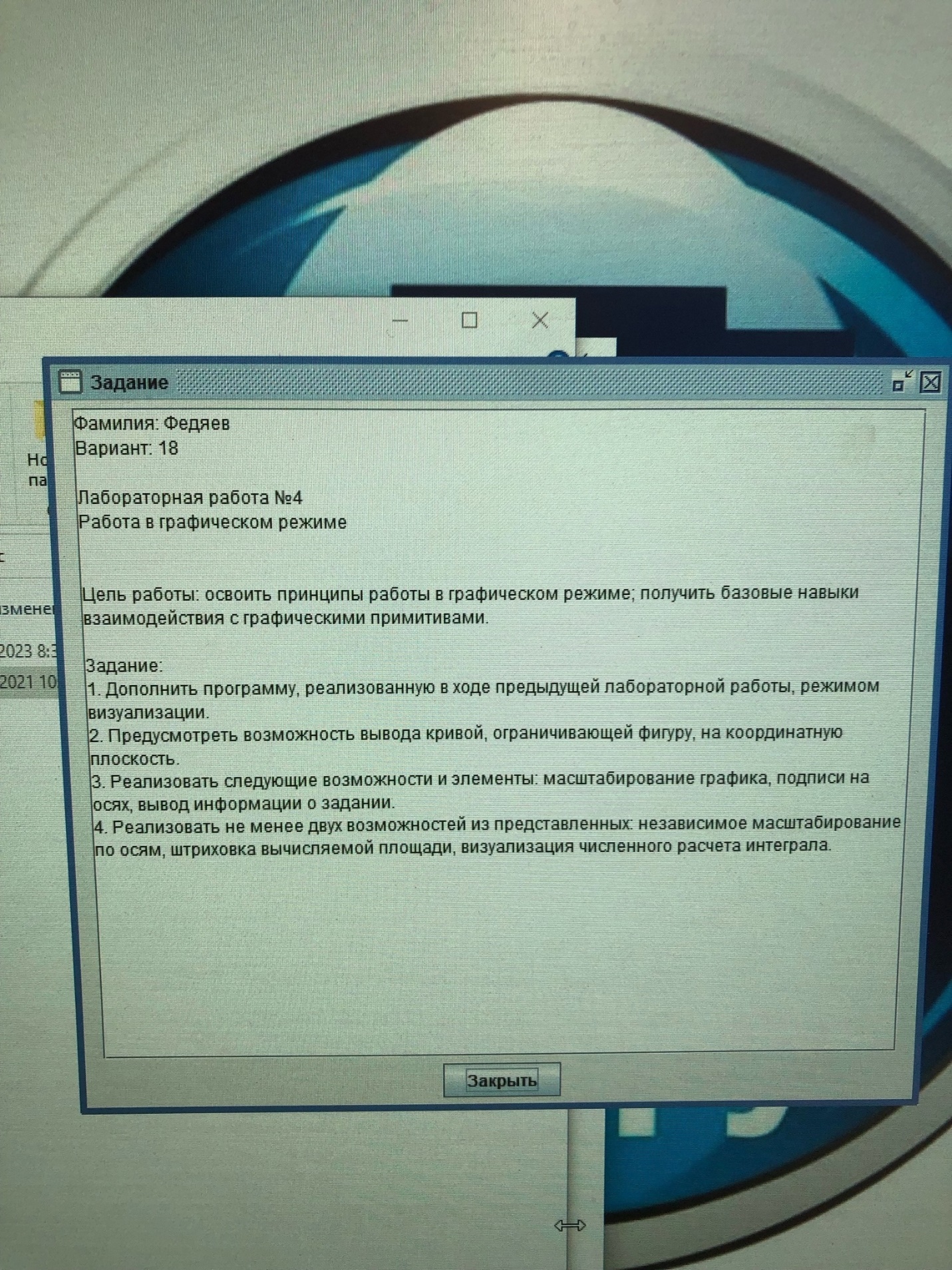
Киров

2023

**Цель работы**

Цель работы: освоить принципы работы в графическом режиме, получить базовые навыки взаимодействия с графическими примитивами.

**Формулировка задания**



**Описание алгоритма первой программы**

Необходимо посчитать площадь фигуры с помощью метода средних прямоугольников, погрешность, определённый интеграл.

1. Создаём функцию f(x), которая вычисляет функцию введённого x;
2. Создаём функции integral, которая вычисляет интеграл введённого значения;
3. Создаём функцию square, которая вычисляет площадь разностью пределов интегрирования;
4. Создаём функцию LeftRect, с помощью неё вычисляем определённый интеграл с применением метода средних прямоугольников;
5. Создаём две процедуры: pogreshnost (выводит значение погрешности) и new\_predel (вводим пределы интегрирования и количество прямоугольников).
6. Создаём функцию graph, rects and draw\_plane для вывода графика, кривой и количества треугольников.
7. Создаём case-меню, с помощью которых пользователь будет запускать функции и процедуры.

**Схема алгоритма первой программы**

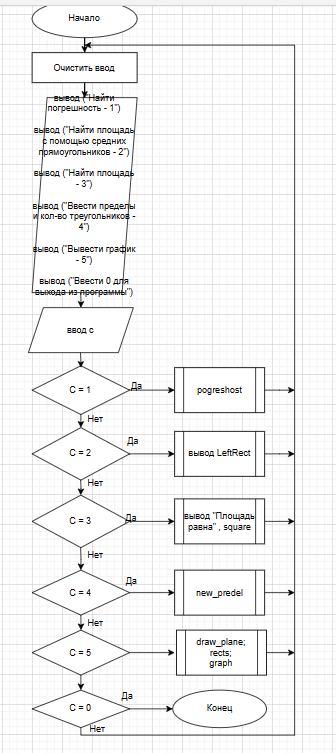


Рисунок 1 – Схема алгоритм задачи

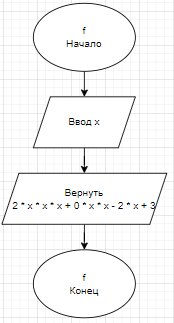


Рисунок 2 – схема алгоритма функции f

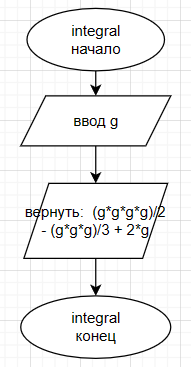


Рисунок 3 – схема алгоритма функции integral

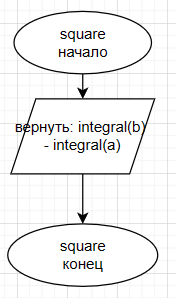
****

Рисунок 4 – схема алгоритма функции square

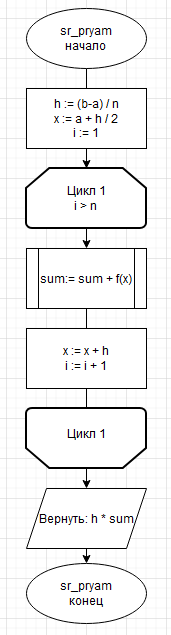


Рисунок 5 – схема алгоритма функции LeftRect

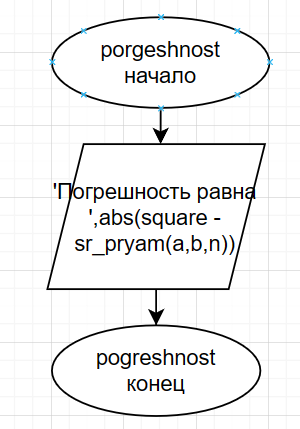
****

Рисунок 6 – схема алгоритма процедуры pogreshnost

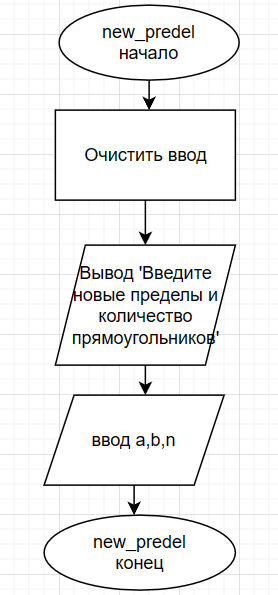


Рисунок 7 – схема алгоритма процедуры new\_predel

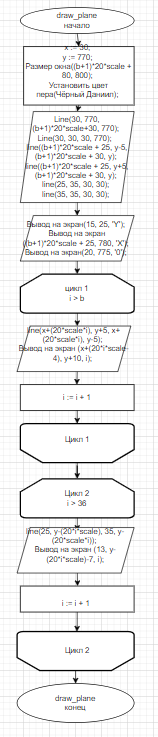


Рисунок 8 – Схема алгоритма функции draw\_plane

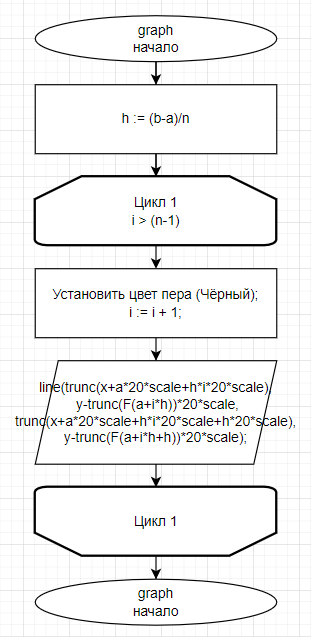


Рисунок 9 – Схема алгоритма функции graph

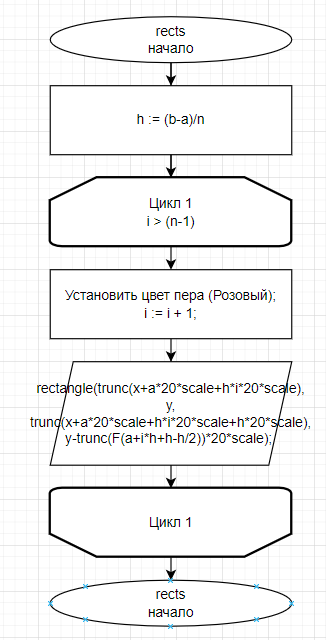


Рисунок 10 – Схема алгоритм функции rects

**Код программы**

**uses** CRT,GraphABC;

**var** i,c,a,b,n,scale,y,x : integer; integralsum : real;

**function** F(x: Real): Real;

**begin**

Result := 2 \* x \* x \* x + 0 \* x \* x - 2 \* x + 3;

**end**;

**function** integral(g : real) : real;

**begin**

Result := (g\*g\*g\*g)/2 - (g\*g\*g)/3 + 2\*g

**end**;

**procedure** draw\_plane(scale, a, b:integer);

**var**

i, x, y: integer;

**begin**

x := 30;

y := 770;

SetWindowSize((b+1)\*20\*scale + 80, 800);

SetPenColor(clBlack);

Line(30, 770, (b+1)\*20\*scale+30, 770); // Ось X

Line(30, 30, 30, 770); // Ось Y

line((b+1)\*20\*scale + 25, y-5, (b+1)\*20\*scale + 30, y);

line((b+1)\*20\*scale + 25, y+5, (b+1)\*20\*scale + 30, y);

line(25, 35, 30, 30);

line(35, 35, 30, 30);

TextOut(15, 25, 'Y');

TextOut((b+1)\*20\*scale + 25, 780, 'X');

TextOut(20, 775, '0');

**for** i:= 1 **to** b **do**

**begin**

line(x+(20\*scale\*i), y+5, x+(20\*scale\*i), y-5);

TextOut(x+(20\*i\*scale-4), y+10, i);

**end**;

**for** i:= 1 **to** 36 **do**

**begin**

line(25, y-(20\*i\*scale), 35, y-(20\*scale\*i));

TextOut(13, y-(20\*i\*scale)-7, i);

**end**;

**end**;

**function** graph(a, b, n, x, y, scale:integer):boolean;

**var**

i: integer;

h: Real;

**begin**

h := (b - a) / n;

**for** i:= 0 **to** n-1 **do**

**begin**

SetPenColor(clBlack);

line(trunc(x+a\*20\*scale+h\*i\*20\*scale), y-trunc(F(a+i\*h))\*20\*scale, trunc(x+a\*20\*scale+h\*i\*20\*scale+h\*20\*scale), y-trunc(F(a+i\*h+h))\*20\*scale);

**end**;

**end**;

**function** rects(a, b, n, x, y, scale:integer):boolean;

**var**

i: integer;

h: Real;

**begin**

h := (b - a) / n;

**for** i:=0 **to** n-1 **do**

**begin**

SetBrushColor(clPink);

rectangle(trunc(x+a\*20\*scale+h\*i\*20\*scale), y, trunc(x+a\*20\*scale+h\*i\*20\*scale+h\*20\*scale), y-trunc(F(a+i\*h+h-h/2))\*20\*scale);

**end**;

**end**;

**function** square : real;

**begin**

Result := integral(b) - integral(a);

**end**;

**function** LeftRect(a, b: Real; n: Integer): Real;

**var**

h, x, sum: Real;

i: Integer;

**begin**

h := (b - a) / n;

sum := 0;

**for** i := 0 **to** n - 1 **do**

**begin**

x := a + h \* i;

sum := sum + F(x);

**end**;

LeftRect := h \* sum;

**end**;

**procedure** pogreshnost;

**begin**

Writeln('Погрешность равна ',abs(square - LeftRect(a,b,n)));

readln();

**end**;

**procedure** new\_predel;

**begin**

ClrScr;

writeln('Введите новые пределы и количество прямоугольников: ');

readln(a,b,n);

**end**;

**begin**

x := 30;

y := 770;

scale := 1;

SetConsoleIO;

**repeat**

ClrScr;

writeln('Найти погрешность - 1');

writeln('Найти площадь с помощью левых прямоугольников - 2');

writeln('Найти площадь - 3');

writeln('Ввести пределы и количество прямоугольников- 4');

writeln('Вывести график - 5');

writeln('Ввести 0 для выхода из программы');

readln(c);

**case** c **of**

1 : pogreshnost;

2 : **begin** print(LeftRect(a,b,n)); **var** ch : char; **repeat** readln(ch) **until** ch = #13; **end**;

3 : **begin** write('Площадь равна '); writeln(square); readln(); **end**;

4 : new\_predel;

5 :

**begin**

draw\_plane(scale, a, b);

rects(a, b, n, x, y, scale);

graph(a, b, n, x, y, scale);

**end**;

**end**;

**until** c = 0;

**end**.

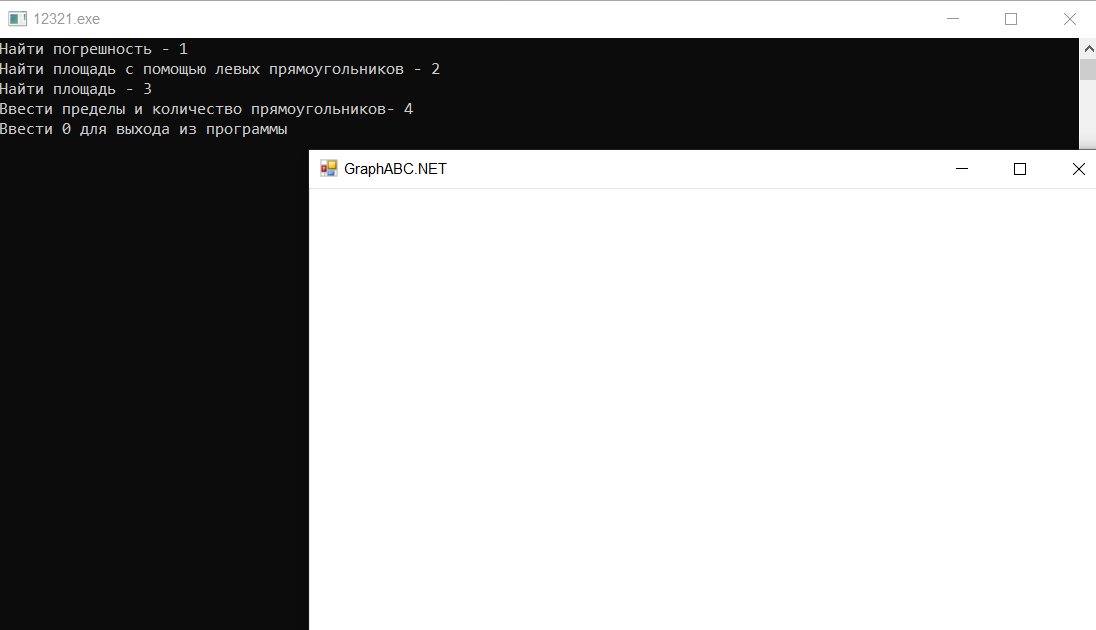
****

Рисунок 11 – Результат выполнения программы 1

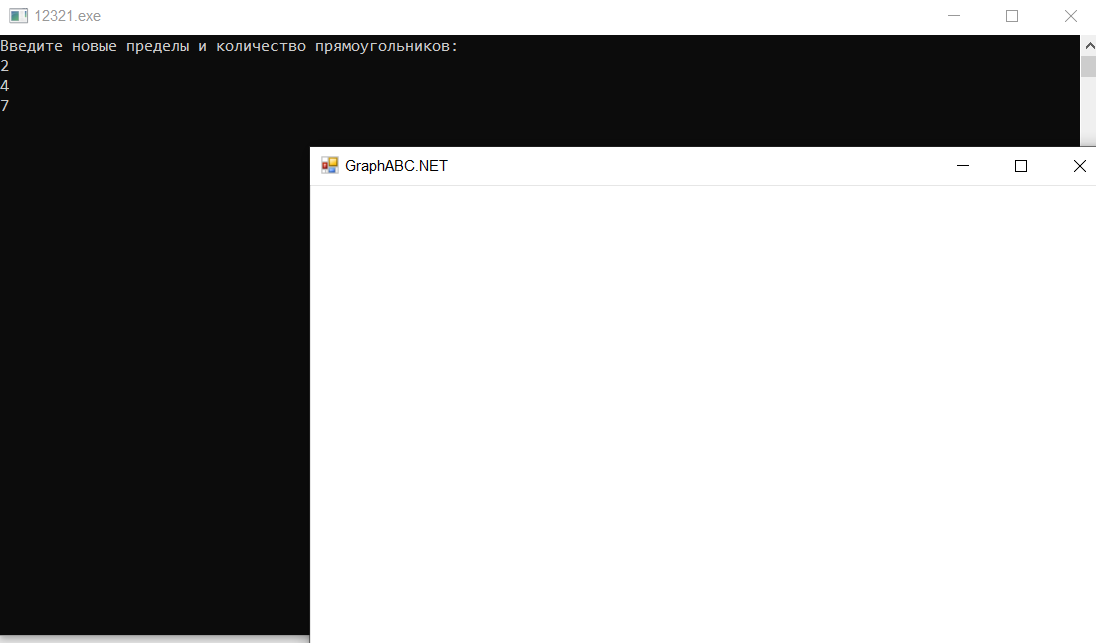


Рисунок 12 – Результат выполнения программы 2

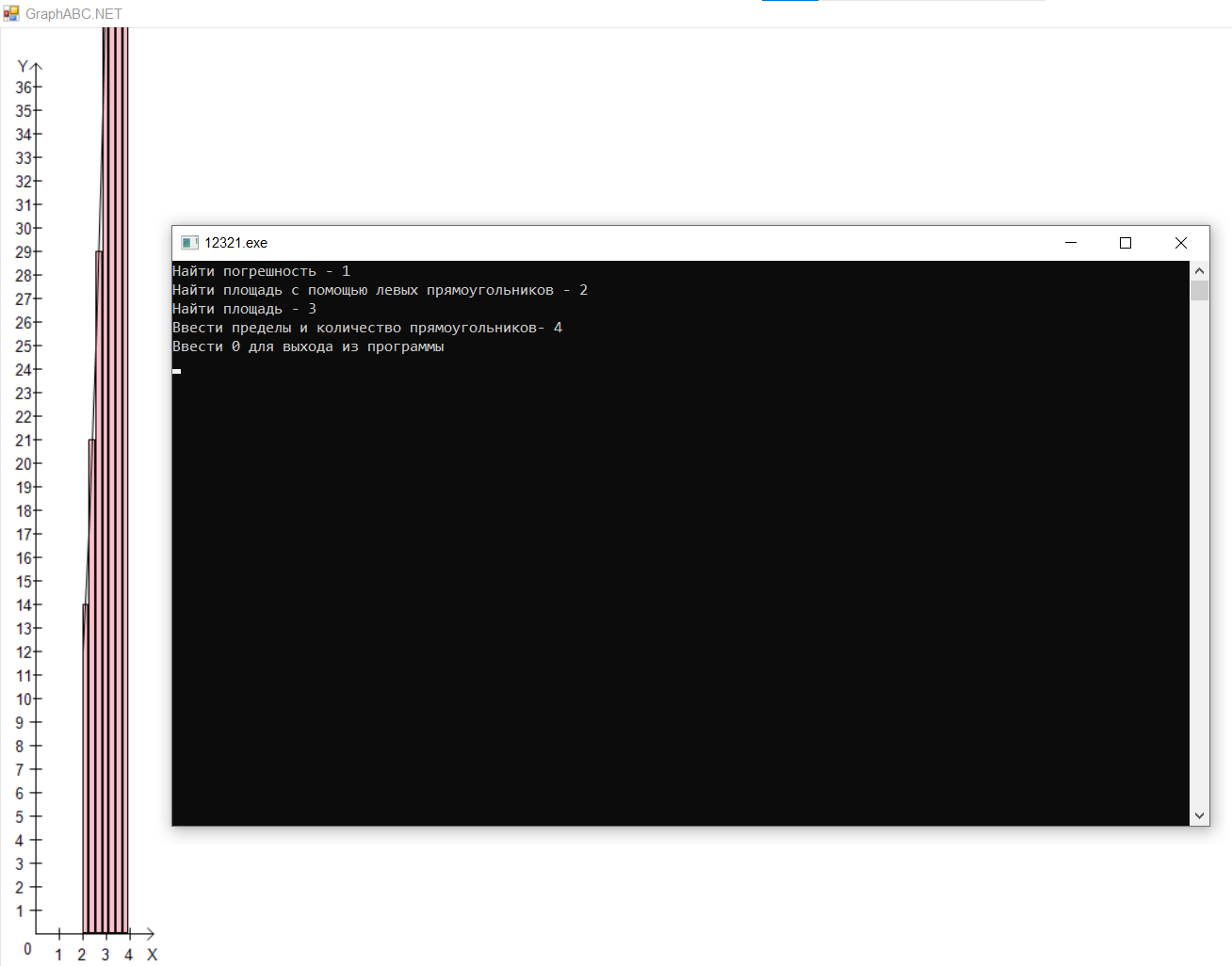
****

Рисунок 13 – Результат выполнения программы 3

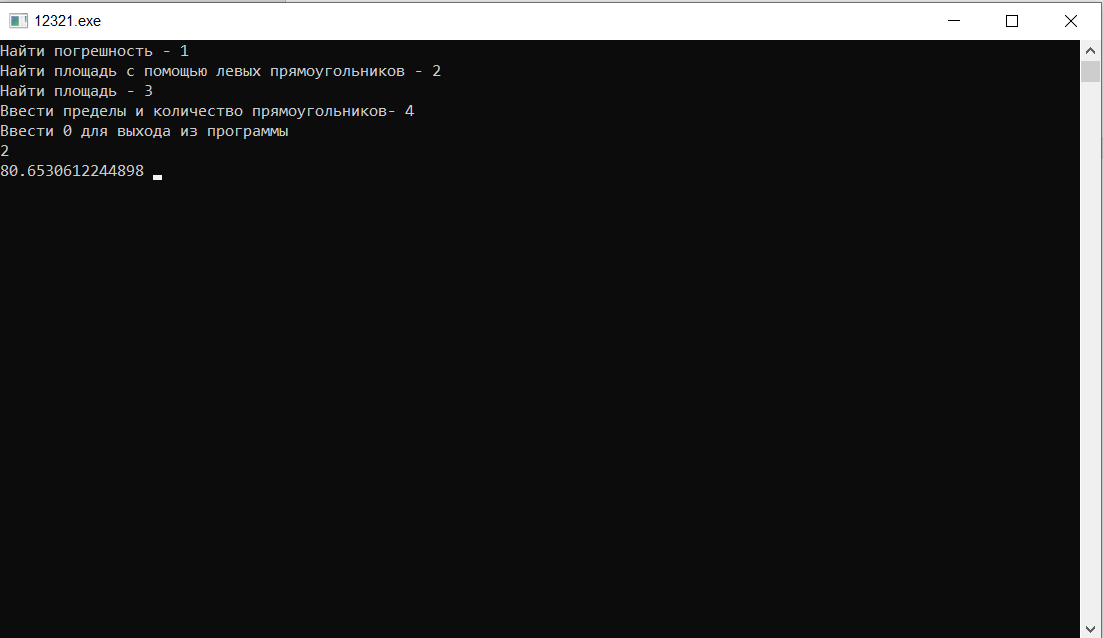


Рисунок 14 – Результат выполнения программы 4

**Вывод**

В ходе выполнения домашней лабораторной работы я закрепили знания, полученные при выполнение лабораторной работы №9-12. При написании кода задачи были использованы конструкции case-меню, изученные ранее. При написании и использовании функций возникали сложности с типами данных. Так же на практике была изучена библиотека CRT и GraphABC. Научились работать в графическом режиме на языке программирования PascalABC. Так же для написания схемы алгоритма использовал draw.io. Его использование упрощает написание схем алгоритмов. Опыт написания схем-алгоритмов будет полезен в будущем для решения новых задач а так же понимание программ, написанных другими пользователями