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»**

**ПО ДИСЦИПЛИНЕ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»**

Выполнила: студентка учебной группы

ИСПк-203-52-00

Бочкарёва Виктория Дмитриевна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**Цель работы**

Освоить принцип работы в графическом режиме, получить базовые навыки взаимодействия с графическими примитивами.

**Формулировка задания**

Фамилия: Бочкарева

Вариант: 3

Задание:

1. Дополнить программу, реализованную в ходе предыдущей лабораторной работы, режимом визуализации.
2. Предусмотреть возможность вывода кривой, ограничивающей фигуру, на координатную плоскость.
3. Реализовать следующие возможности и элементы: масштабирование графика, подписи на осях, вывод информации о задании.
4. Реализовать не менее двух возможностей из представленных: независимое масштабирование по осям, штриховка вычисляемой площади, визуализация численного расчета интеграла.

**Описание алгоритма**

Чтобы дополнить программу из предыдущей лабораторной работы, необходимо подключить модуль *GraphABC*. Для переключения между консолью и графическим окном используются процедуры *SetConsoleIO* и *SetGraphABCIO.* Для работы с графическими примитивами используются процедуры *SetPixel* (закрашивает пиксель), *Line* (рисует линию), *Rectangle* (рисует прямоугольник), *TextOut* (выводит текст в определенном месте графического окна), *SetBrushColor* (устанавливает цвет кисти), *SetBrushStyle* (устанавливает стиль кисти), *SetBrushHatch* (устанавливает стиль штриховки), *SetFontSize* (устанавливает размер выводимого текста), *SetFontColor* (устанавливает цвет выводимого текста), *SetWindowSize* (устанавливает размеры графического окна).

Чтобы изменить масштаб графика, пользователь вводит параметры графика, такие как крайние значения осей и единичный отрезок по х и по у.

Для отрисовки графика используется цикл, в котором рассчитывается значение функции, данной в предыдущей лабораторной работе, и закрашивает пиксели с координатами х и у.

Если пользователь ввел значения a и b, то на график добавляются прямые x = a и x = b, ограничивающие кривую.

С помощью цикла в промежутке от a до b рисуются прямоугольники по заданному пользователем количеству промежутков.

В результате выполнения программы, при выборе пункта «Посмотреть график», в консоли пользователь вводит параметры для масштабирования графика, а в графическом окне выводится криволинейная трапеция в системе координат и визуализация расчета площади фигуры по методу левых прямоугольников, а также информация о результате интегрирования.

**Схема алгоритма**

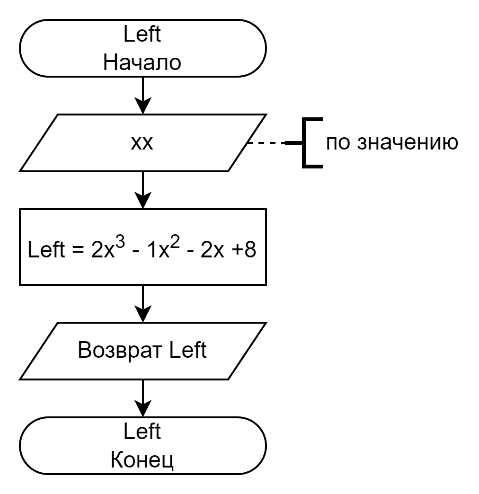


Рис. 1 – Схема функции для вычисления площади криволинейной трапеции по правилу левых прямоугольников

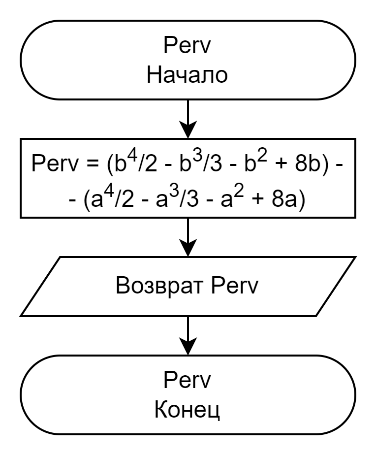


Рис. 2 – Схема функции для вычисления площади криволинейной трапеции по формуле Ньютона-Лейбница

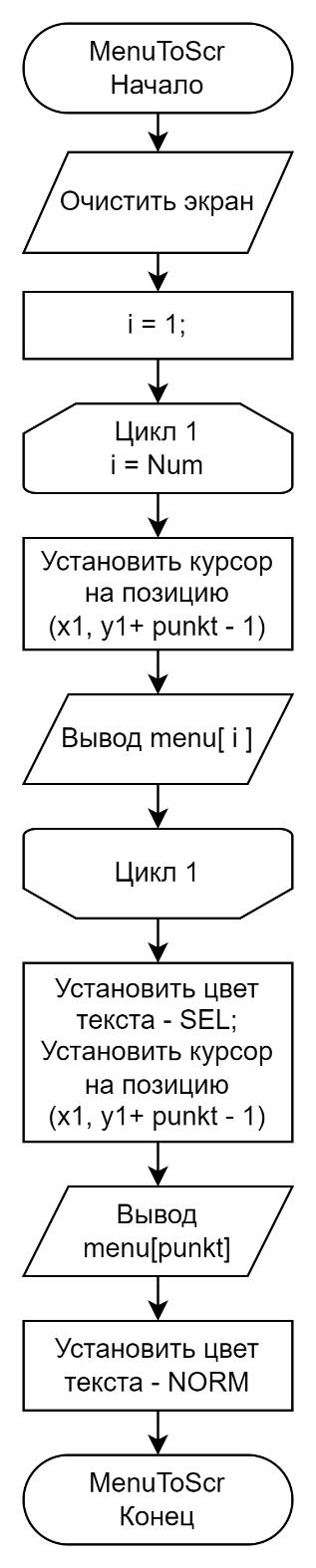


Рис. 3 – Схема процедуры для вывода меню на экран



Рис. 4 – Схема процедуры для вывода информации о задании в графическое окно

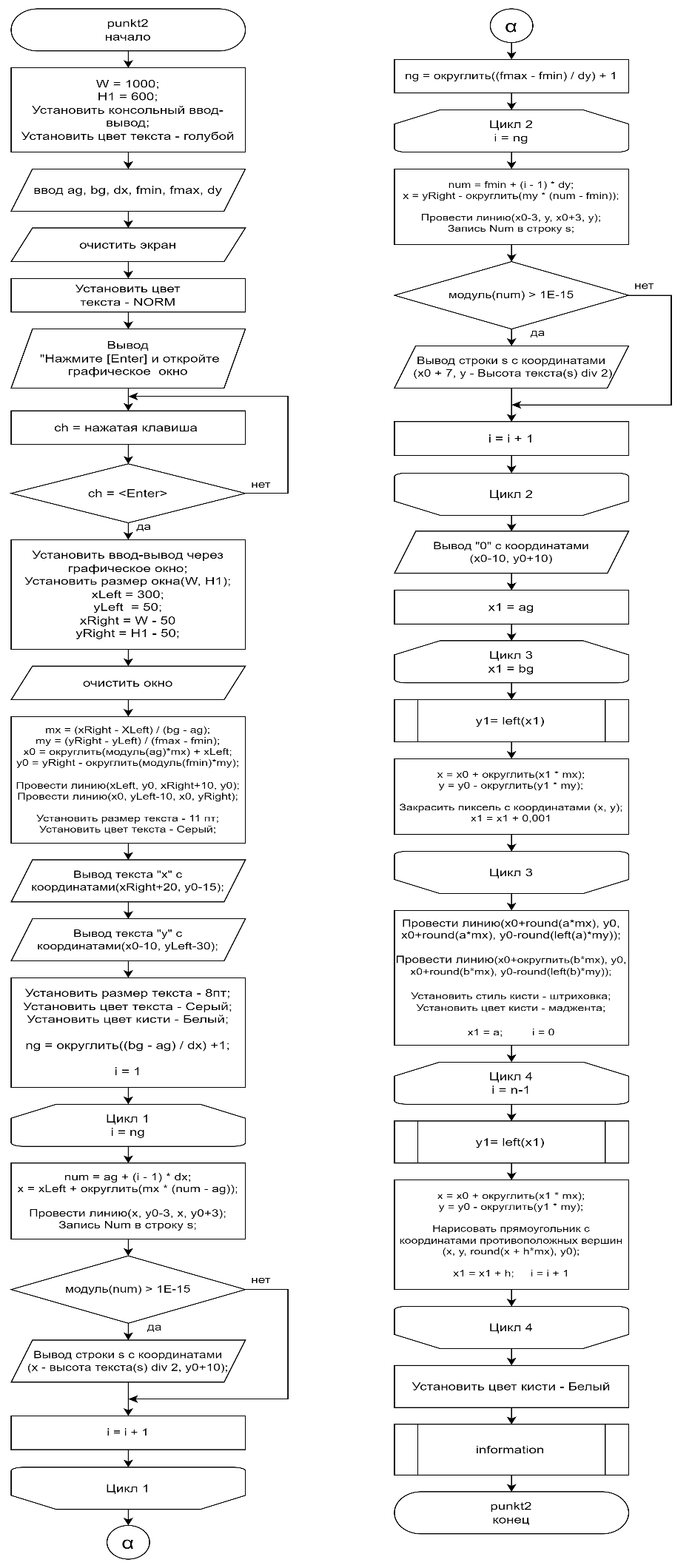


Рис. 5 – Схема процедуры для отрисовки графика

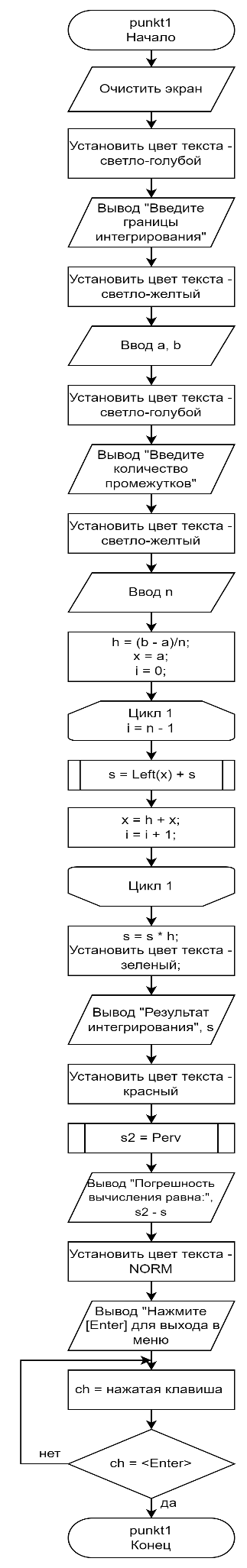


Рис. 6 – Схема процедуры для вычисления площади и нахождения погрешности

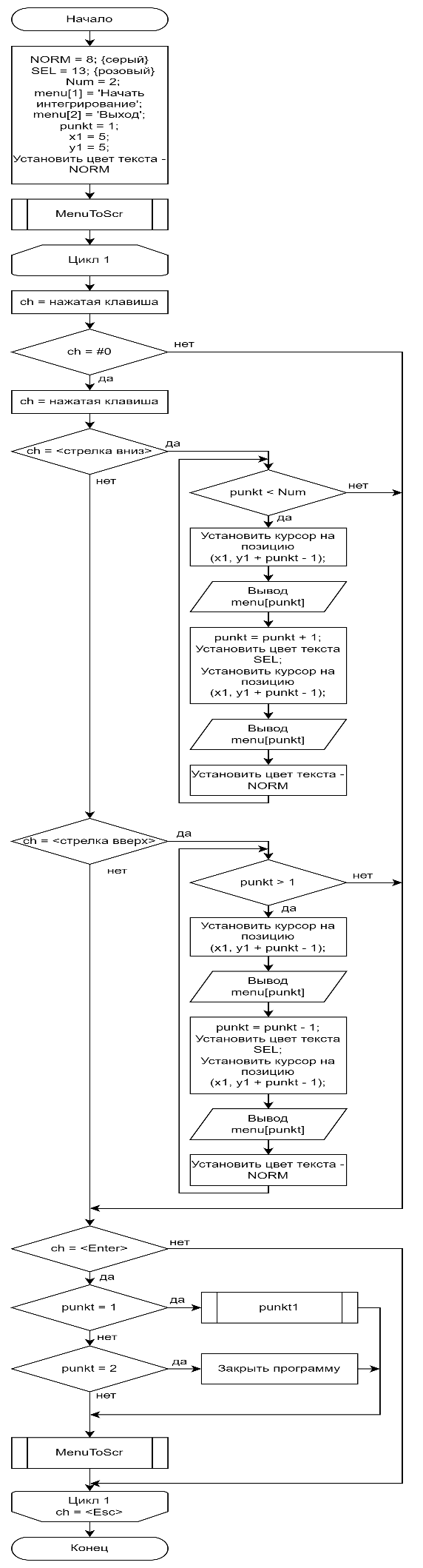


Рис. 7 – Схема алгоритма главной программы

**Код программы**

**program** DKR\_3;

**uses** Crt, GraphAbc;

**const**

NORM = 8; {серый}

SEL = 13; {розовый}

Num = 3;

**var**

menu: **array**[1..Num] **of** string;{ названия пунктов меню }

punkt: integer; ch: char; x1, y1: integer; n: integer;

a, b, x, s, h, dx, dy: real;

**function** Left(xx: real): real;

**begin**

Left := 2 \* (xx \* xx \* xx) + (-1) \* (xx \* xx) + (-2) \* xx + 8;

**end**;

**function** Perv: real;

**begin**

Perv := ((b \*\* 4) / 2 - (b \*\* 3) / 3 - b \*\* 2 + 8 \* b) - ((a \*\* 4) / 2 - (a \*\* 3) / 3 - a \*\* 2 + 8 \* a);

**end**;

**procedure** punkt1;

**begin**

SetConsoleIO;

ClrScr;

textcolor(11);

Writeln('Введите границы интегрирования: ');

textcolor(14);

readln(a, b);

textcolor(11);

Writeln('Введите количество промежутков: ');

textcolor(14);

readln(n);

h := (b - a) / n;

x := a;

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

**begin**

s += Left(x);

x += h;

**end**;

s \*= h;

textcolor(10);

writeln('Результат интегрирования: ', s:0:2);

textcolor(12);

writeln;

writeln('Погрешность вычисления равна: ', perv - s);

textcolor(norm);

writeln('Нажмите [Enter] для выхода в меню');

**repeat**

ch := readkey;

**until** ch = #13;

**end**;

**procedure** information;

**begin**

setfontsize(11);

setfontcolor(clblack);

**if not** ((a = 0) **and** (b = 0)) **then**

**begin**

writeln('Нижний предел: ', a);

writeln('Верхний предел: ', b);

writeln('Площадь заштрихованной фигуры: ', s:0:2);

writeln('Погрешность: ', perv-s);

**end**

**else** writeln('Вы не ввели данные для интегрирования :(');

**end**;

**procedure** punkt2;

**const**

W = 1000; H1 = 600;//Размеры графического окна

**var**

x0, y0, x, y, xLeft, yLeft, xRight, yRight, ng: integer;

ag, bg, fmin, fmax, x1, y1, mx, my, num: real;

i: byte;

s: string;

**begin**

SetConsoleIO;

textcolor(11);

clrscr;

Writeln('Введите нижнюю границу системы координат по Х: ');

read(ag);

Writeln('Введите верхнюю границу системы координат по Х: ');

read(bg);

Writeln('Введите единичный отрезок по Х: ');

read(dx);

Writeln('Введите нижнюю границу системы координат по Y: ');

read(fmin);

Writeln('Введите верхнюю границу системы координат по Y: ');

read(fmax);

Writeln('Введите единичный отрезок по Y: ');

read(dy);

writeln;

clrscr;

textcolor(norm);

Writeln('Нажмите [Enter] и откройте графическое окно');

**repeat**

ch := readkey;

**until** ch = #13;

SetGraphabcIO;

SetWindowSize(W, H1); //Устанавливаем размеры графического окна

xLeft := 300;

yLeft := 50;

xRight := W - 50;

yRight := H1 - 50;

clearwindow;

mx := (xRight - xLeft) / (bg - ag); //масштаб по Х

my := (yRight - yLeft) / (fmax - fmin); //масштаб по Y

x0 := trunc(abs(ag) \* mx) + xLeft;

y0 := yRight - trunc(abs(fmin) \* my);

line(xLeft, y0, xRight + 10, y0); //ось ОХ

line(x0, yLeft - 10, x0, yRight); //ось ОY

SetFontSize(11);

SetFontColor(clSlateGray);

TextOut(xRight + 20, y0 - 15, 'х');

TextOut(x0 - 10, yLeft - 30, 'у');

SetFontSize(8);

SetFontColor(clgray);

setbrushcolor(clwhite);

ng := round((bg - ag) / dx) + 1; //количество засечек по ОХ

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

**begin**

num := ag + (i - 1) \* dx; //Координата на оси ОХ

x := xLeft + trunc(mx \* (num - ag));

Line(x, y0 - 3, x, y0 + 3);

str(Num:0:1, s);

**if** abs(num) > 1E-15 **then** //Исключаем 0 на оси OX

TextOut(x - TextWidth(s) **div** 2, y0 + 10, s)

**end**;

ng := round((fmax - fmin) / dy) + 1; //количество засечек по ОY

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

**begin**

num := fMin + (i - 1) \* dy; //Координата на оси ОY

y := yRight - trunc(my \* (num - fmin));

Line(x0 - 3, y, x0 + 3, y);

str(num:0:0, s);

**if** abs(num) > 1E-15 **then** //Исключаем 0 на оси OY

TextOut(x0 + 7, y - TextHeight(s) **div** 2, s)

**end**;

TextOut(x0 - 10, y0 + 10, '0'); //Нулевая точка

x1 := ag;

**while** x1 <= bg **do**

**begin**

y1 := left(x1);

x := x0 + round(x1 \* mx);

y := y0 - round(y1 \* my);

SetPixel(x, y, clred);

x1 := x1 + 0.001

**end**;

line(x0 + round(a\*mx), y0, x0 + round(a\*mx), y0 - round(left(a)\*my), clblue); // х = а

line(x0 + round(b\*mx), y0, x0 + round(b\*mx), y0 - round(left(b)\*my), clblue); // х = b

setbrushstyle(bsHatch);

setbrushhatch(bhLargeConfetti);

setbrushcolor(clMagenta);

x1:=a;

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

**begin**

y1:= left(x1);

x:= x0 + round(x1 \* mx);

y:= y0 - round(y1 \* my);

rectangle(x, y, round(x + h\*mx), y0);

x1:= x1 + h;

**end**;

setbrushcolor(clWhite);

information;

**end**;

**procedure** MenuToScr;

**var**

i: integer;

**begin**

SetConsoleIO;

ClrScr;

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

**begin**

GoToXY(x1, y1 + i - 1);

write(menu[i]);

**end**;

TextColor(SEL);

GoToXY(x1, y1 + punkt - 1);

write(menu[punkt]);

TextColor(NORM);

**end**;

**begin**

SetConsoleIO;

clrscr;

menu[1] := ' Начать интегрирование ';

menu[2] := ' Показать график ';

menu[3] := ' Выход ';

punkt := 1; x1 := 5; y1 := 5;

TextColor(NORM);

MenuToScr;

**repeat**

ch := ReadKey;

**if** ch = #0 **then begin**

ch := ReadKey;

**case** ch **of**

#40:{ стрелка вниз }

**if** punkt < Num **then begin**

GoToXY(x1, y1 + punkt - 1); write(menu[punkt]);

punkt := punkt + 1;

TextColor(SEl);

GoToXY(x1, y1 + punkt - 1); write(menu[punkt]);

TextColor(NORM);

**end**;

#38:{ стрелка вверх }

**if** punkt > 1 **then begin**

GoToXY(x1, y1 + punkt - 1); write(menu[punkt]);

punkt := punkt - 1;

TextColor(SEl);

GoToXY(x1, y1 + punkt - 1); write(menu[punkt]);

TextColor(NORM);

**end**;

**end**;

**end**

**else**

**if** ch = #13 **then begin**{ нажата клавиша <Enter> }

**case** punkt **of**

1: punkt1;

2: punkt2;

3: CloseWindow;{ выход }

**end**;

MenuToScr;

**end**;

**until** ch = #27;{ 27 - код <Esc> }

**end**.

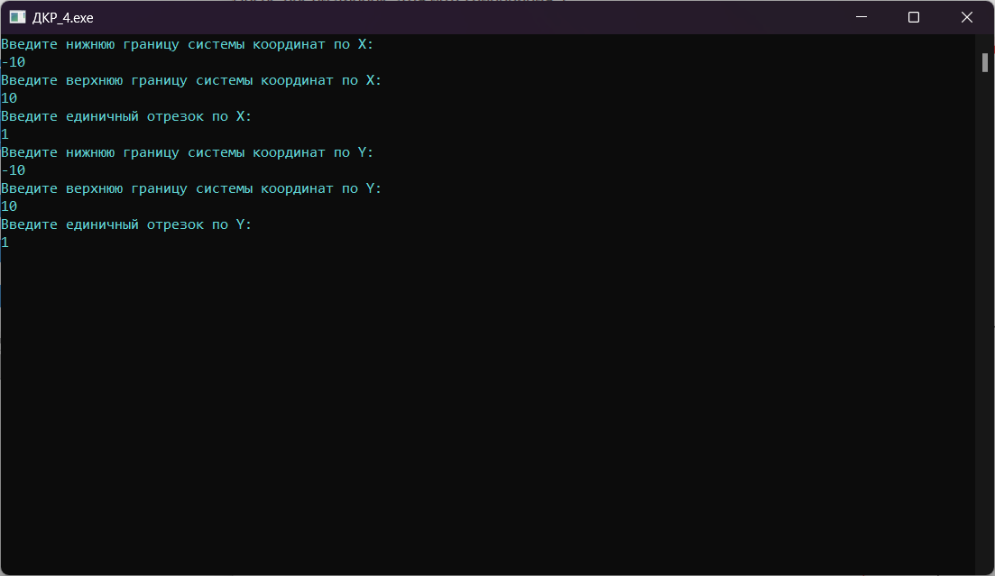
**Результат выполнения программы**

Рис. 8 – Ввод данных для масштабирования графика

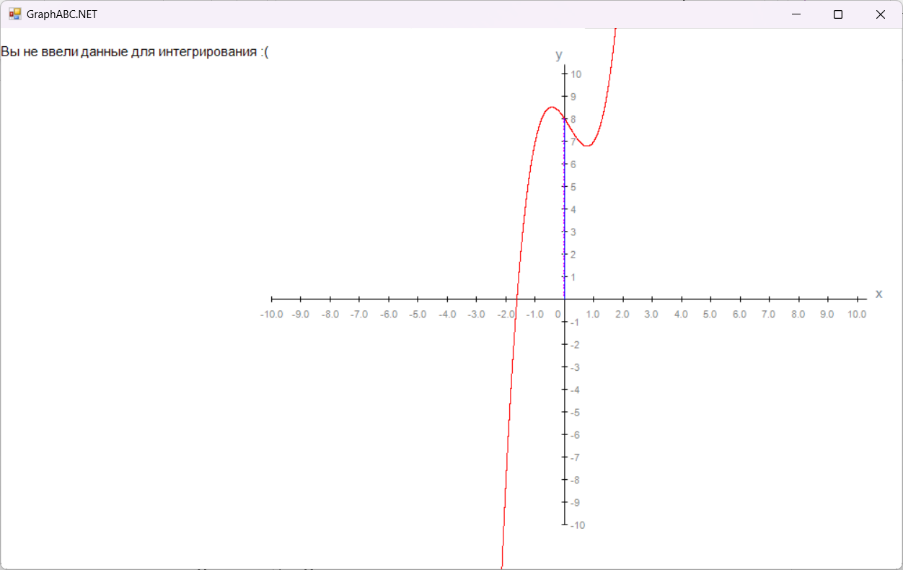


Рис. 9– График без предварительного интегрирования

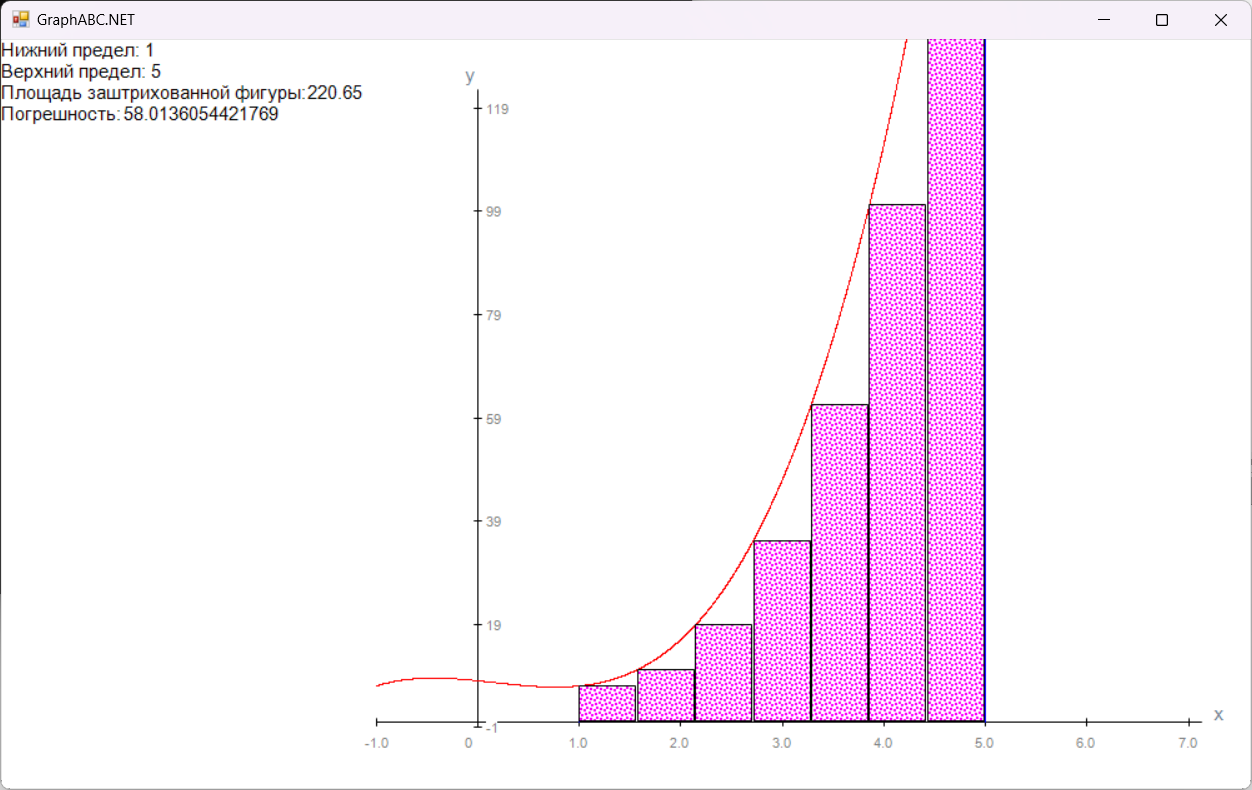


Рис. 10 – График с предварительным интегрированием

**Вывод**

В результате выполнения домашней контрольной работы поставленная цель была достигнута. Был освоен принцип работы в графическом режиме, получены базовые навыки взаимодействия с графическими примитивами.

По мере выполнения заданий, серьезных проблем не возникало, но были небольшие трудности, так, например, масштабирование графика. Изначально масштаб графика был задан константой, и, соответственно, единичный отрезок, а также крайние значения осей, не могли изменяться, поэтому было решено сделать их переменными, чтоб пользователь вводил их по своему усмотрению. Также сложность вызвало то, что при подключении модуля GraphABC, при запуске программы, все, что должно быть в консоли, выводилось в графическое окно, и, чтобы не переписывать половину кода, были использованы процедуры SetConsoleIO и SetGraphABCIO, которые позволяют перенаправлять ввод-вывод в зависимости от цели.

Таким образом, справившись с возникнувшими проблемами, а также получив новые знания, была достигнута поставленная цель, выполнены задачи, получены удовлетворяющие результаты.