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**

**«ИЗУЧЕНИЕ БАЗОВЫХ ПРИНЦИПОВ ОРГАНИЗАЦИИ ПРОЦЕДУР И ФУНКЦИЙ»**

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

Выполнил: студент учебной группы ИСПк- 204-52-00

Алиев Тимур Русланович

Преподаватель:

Сергеева Елизавета Григорьевна

г. Киров

2023

1. **Цель работы**: освоить принципы работы в графическом режиме, получить базовые навыки взаимодействия с графическими примитивами.
2. **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 расчёта интеграла.

1. **Описание алгоритма:**

Для решения данной задачи используется модуль CRT для отображения консоли на экране пользователя, которая взаимодействует с самим пользователем. В меню case есть выбор из 2 пунктов, выбор которых осуществляется через case... выбор оператора. Чтобы перемещаться по меню, используйте процесс GoToXY для перемещения курсора в указанное положение на экране. Функция ReadKey используется для считывания клавиш с клавиатуры. Для визуального оформления контекстного меню используйте процесс textColor, чтобы изменить цвет текста.

Алгоритм использует метод Симпсона для вычисления определенного интеграла. Алгоритм использует функции и процедуры для вычисления производных, примитивов и фиксированных интегралов.

Когда будет выбрана опция «Начать интегрирование», пользователю будет предложено ввести предел интегрирования и количество интервалов, чтобы изменить точность вычисления конкретного интегрирования.

Для создания визуализации графиков функций использовался модуль GraphABC. Для переключения между консолью и графическим окном используются процедуры SetConsoleIO и SetGraphABCIO. Для работы с графическими примитивами используются различные процедуры, такие как SetPixel (для закрашивания пикселей), Line (для рисования линий), Rectangle (для рисования прямоугольников), TextOut (для вывода текста), SetBrushColor (для установки цвета кисти) и другие.

Для масштабирования графика пользователь задает параметры графика, границы осей и единичный отрезок по x и у. Затем при отрисовке графика в цикле рассчитываются значения функции и закрашиваются пиксели с соответствующими координатами.

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

Также с помощью цикла рисуются прямоугольники в заданном пользователем количестве промежутков в интервале от a до b.

В результате выполнения программы, при выборе пункта "Смотреть график", пользователь вводит параметры для масштабирования графика в консоли, а в графическом окне отображается криволинейная трапеция в системе координат, визуализация расчета площади фигуры по методу правых прямоугольников и информация о результате интегрирования.

1. **Схема алгоритма с комментариями:**

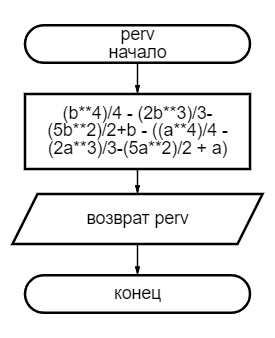
****

Рис. 1 «Схема функции perv»

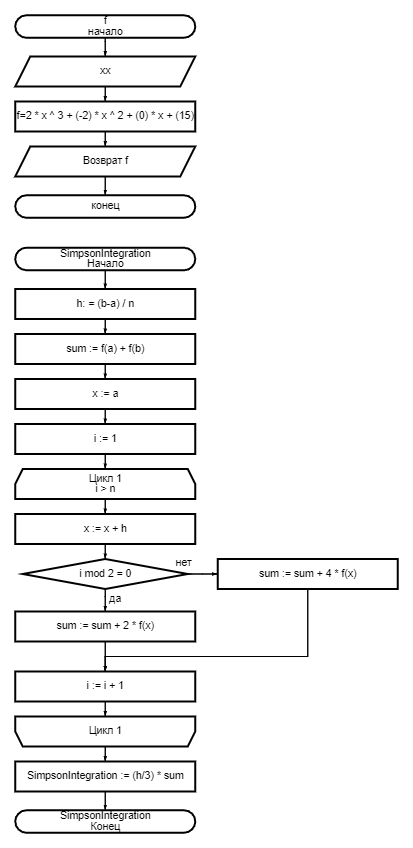
****

Рис. 2 «Схема алгоритма подпрограммы f and SimpsonIntegration»

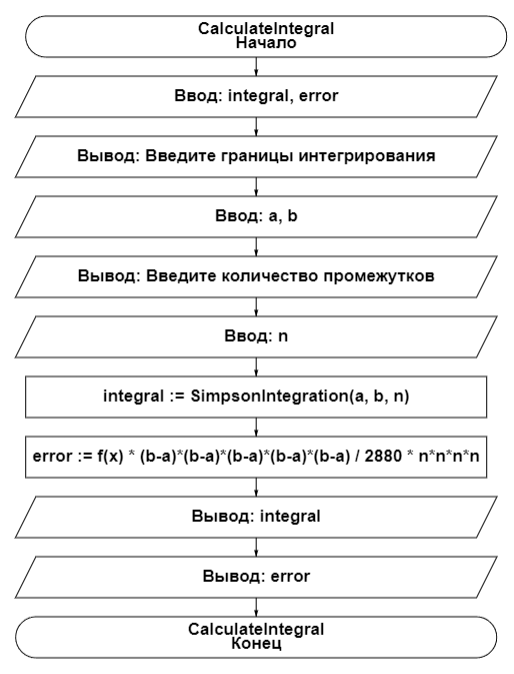


Рис. 3 «Схема алгоритма подпрограммы calculateintegral»

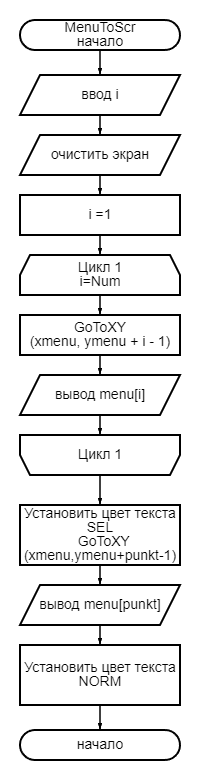
****

Рис. 4 «Схема алгоритма подпрограммы MenuToScr»

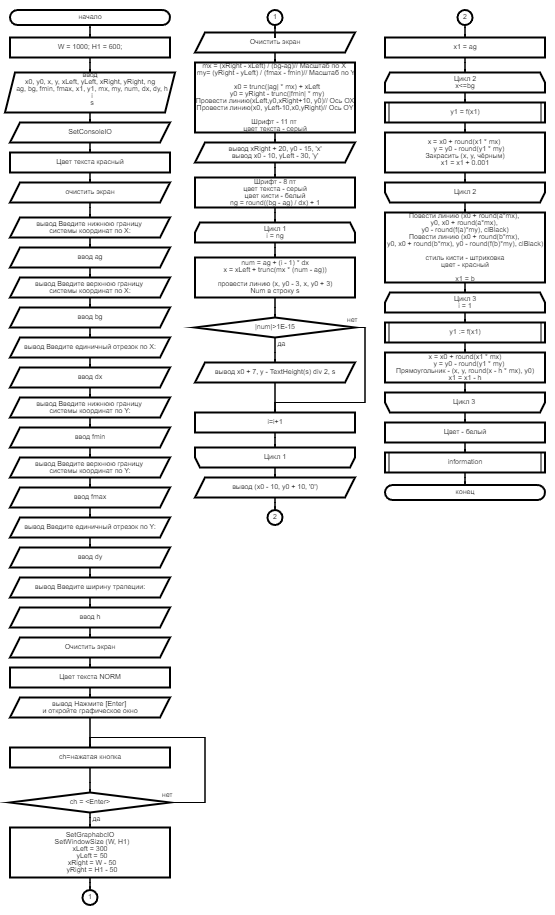
****

Рис. 5 «Схема алгоритма подпрограммы punkt2»

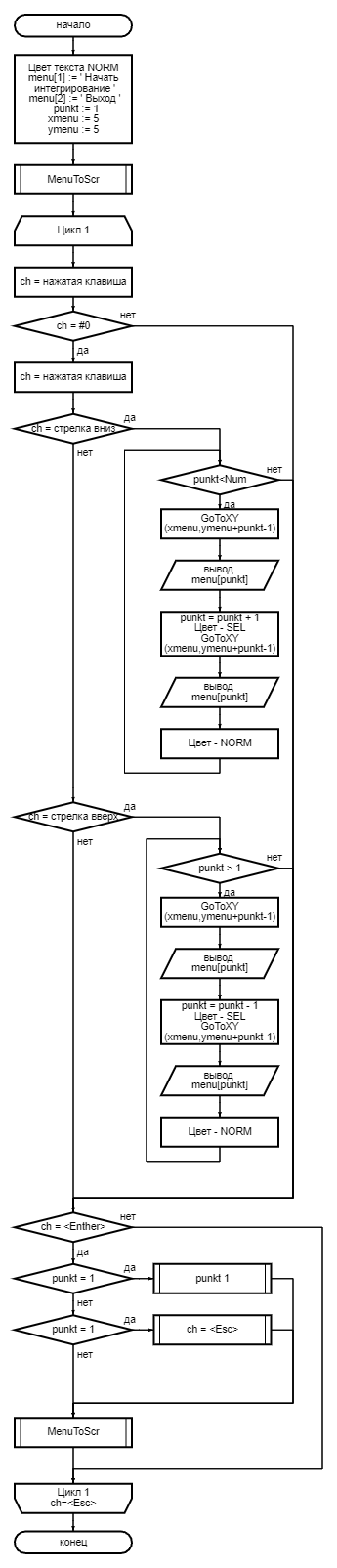
****

Рис. 6 «Схема алгоритма»

1. **Код программы:**

**uses** Crt, GraphABC;

**const**

NORM = 12; // цвет невыделеного пункта

SEL = 10; // цвет выделенного пункта

Num = 3; //3 пункта меню

**var**

menu: **array**[1..Num] **of** string[24];//названия пунктов меню

punkt: integer;

ch: char;

xmenu, ymenu, TextAttr: byte;

a, b, c, x, d, ff, s, dx, dy: real;

n: integer;

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

**begin**

f := 2 \* xx \* xx \* xx + (-2) \* xx \* xx + (0) \* xx + 15;

**end**;

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

**begin**

perv:= ((b\*\*4)/4 - (2\*b\*\*3)/3 - (5\*b\*\*2)/2 + b) - ((a\*\*4)/4 - (2\*a\*\*3)/3 - (5\*a\*\*2)/2 + a);

**end**;

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

**var**

h, x, sum: Real;

i: Integer;

**begin**

h := (b - a) / n;

sum := f(a) + f(b);

x := a;

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

**begin**

x := x + h;

**if** i **mod** 2 = 0 **then**

sum := sum + 2 \* f(x)

**else**

sum := sum + 4 \* f(x);

**end**;

SimpsonIntegration := (h / 3) \* sum;

**end**;

**procedure** calculateIntegral;

**var**

integral, error: real;

**begin**

writeln('Введите границы интегрирования: ');

readln(a, b);

writeln('Введите количество промежутков: ');

readln(n);

integral := SimpsonIntegration(a, b, n);

error := f(x) \* (b-a)\*(b-a)\*(b-a)\*(b-a)\*(b-a) / 2880 \* n\*n\*n\*n;// Вычисление погрешности

writeln('Интеграл равен: ', integral:0:3);

writeln('Погрешность: ', error:0:3);

**end**;

**procedure** punkt1;

**begin**

ClrScr;

calculateIntegral;

writeln;

writeln('Процедура завершена вот такие пироги. Нажмите <Enter> для продолжения.');

**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 фигуры: ', s:0:3);

writeln('Погрешность: ', perv(x)-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, dx, dy, h: real;

i: byte;

s: string;

**begin**

SetConsoleIO;

textcolor(12);

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('Введите ширину трапеции: ');

read(h);

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 := 250;

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

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

line(xLeft, y0, xRight + 10, y0); // Ось OX

line(x0, yLeft - 10, x0, yRight); // Ось OY

SetFontSize(15);

SetFontColor(clSlateGray);

TextOut(xRight + 20, y0 - 15, 'х');

TextOut(x0 - 10, yLeft - 30, 'у');

SetFontSize(10);

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 + round(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 OY

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

**begin**

y := yRight - round(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 := f(x1);

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

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

SetPixel(x+20, y, clBlack);

x1 := x1 + 0.001

**end**;

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

setbrushstyle(bsHatch);

setbrushhatch(bhPercent10);

setbrushcolor(clRed);

x1 := b;

**var** hhru : real := (b-a)/h;

**var** xxx : real := a - hhru;

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

**begin**

xxx := xxx + hhru;

line(x0 + round(a\*mx) + round(xxx\*mx),y0-round(f(xxx)\*my),x0 + round(a\*mx) + round(xxx\*mx),y0);

line(x0 + round(a\*mx) + round(xxx\*mx),y0-round(f(xxx)\*my),x0 + round(a\*mx) + round((xxx+hhru)\*mx),y0-round(f(xxx+hhru)\*my));

**end**;

line(x0 + round(a\*mx) + round((xxx+hhru)\*mx),y0-round(f(xxx +hhru)\*my),x0 + round(a\*mx) + round((xxx + hhru)\*mx),y0);

setbrushcolor(clWhite);

information;

**end**;

**procedure** MenuToScr;// вывод меню на экран

**var**

i: integer;

**begin**

SetConsoleIO;

ClrScr;

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

**begin**

GoToXY(xmenu, ymenu + i - 1);

write(menu[i]);

**end**;

TextColor(SEL);

GoToXY(xmenu, ymenu + punkt - 1);

write(menu[punkt]);// выделим строку меню

TextColor(NORM);

**end**;

**begin**

SetConsoleIO;

ClrScr;

menu[1] := ' Начать интегрирование ';

menu[2] := ' Смотреть график ';

menu[3] := ' Выход ';

punkt := 1; xmenu := 5; ymenu := 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(xmenu, ymenu + punkt - 1); write(menu[punkt]);

punkt := punkt + 1;

TextColor(SEL);

GoToXY(xmenu, ymenu + punkt - 1); write(menu[punkt]);

TextColor(NORM);

**end**;

#38:// стрелка вверх

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

GoToXY(xmenu, ymenu + punkt - 1); write(menu[punkt]);

punkt := punkt - 1;

TextColor(SEL);

GoToXY(xmenu, ymenu + 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;

**end**.

**Результат выполнения программы:**

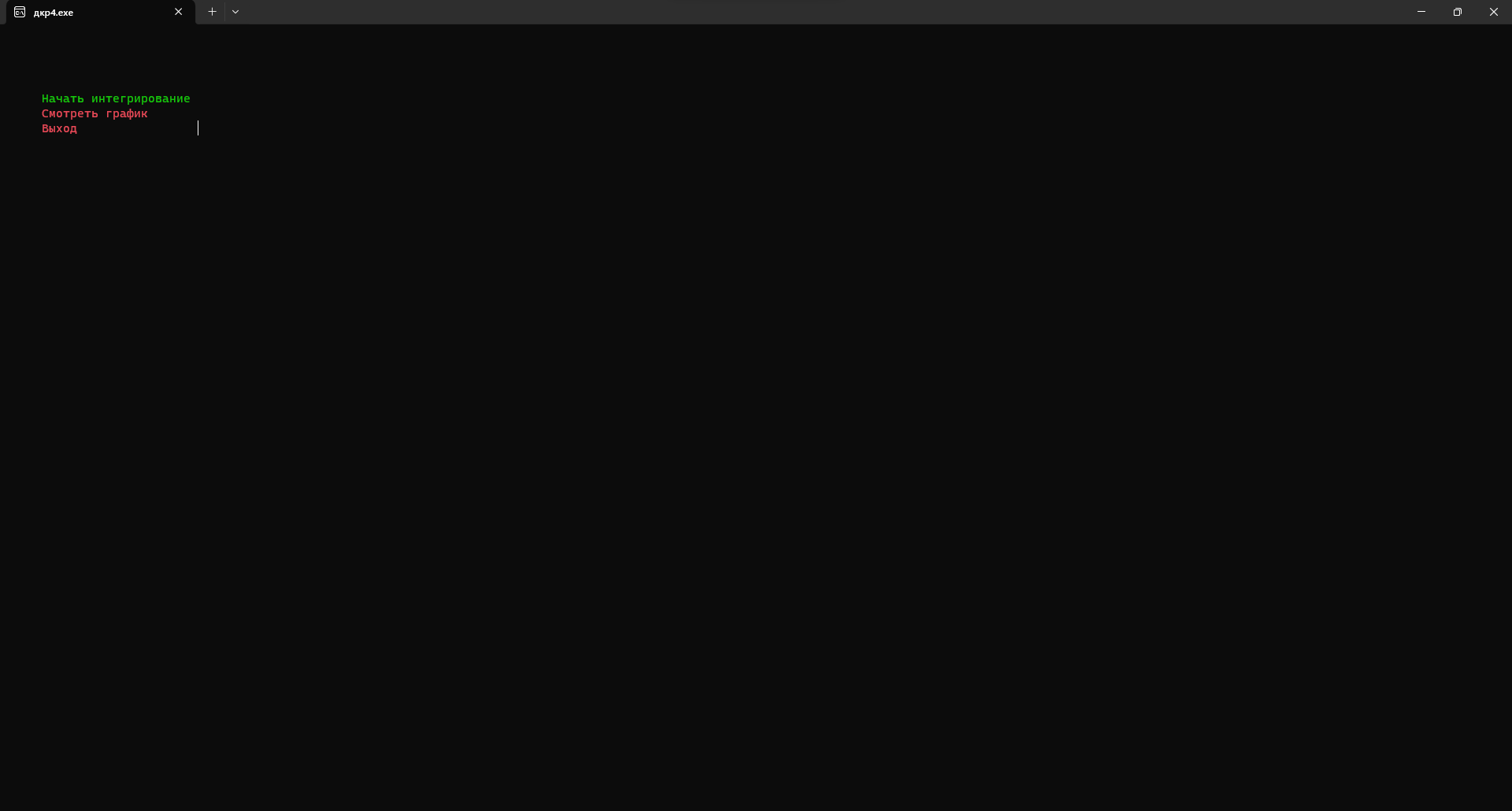
****

Рис. 7 «Case-меню»

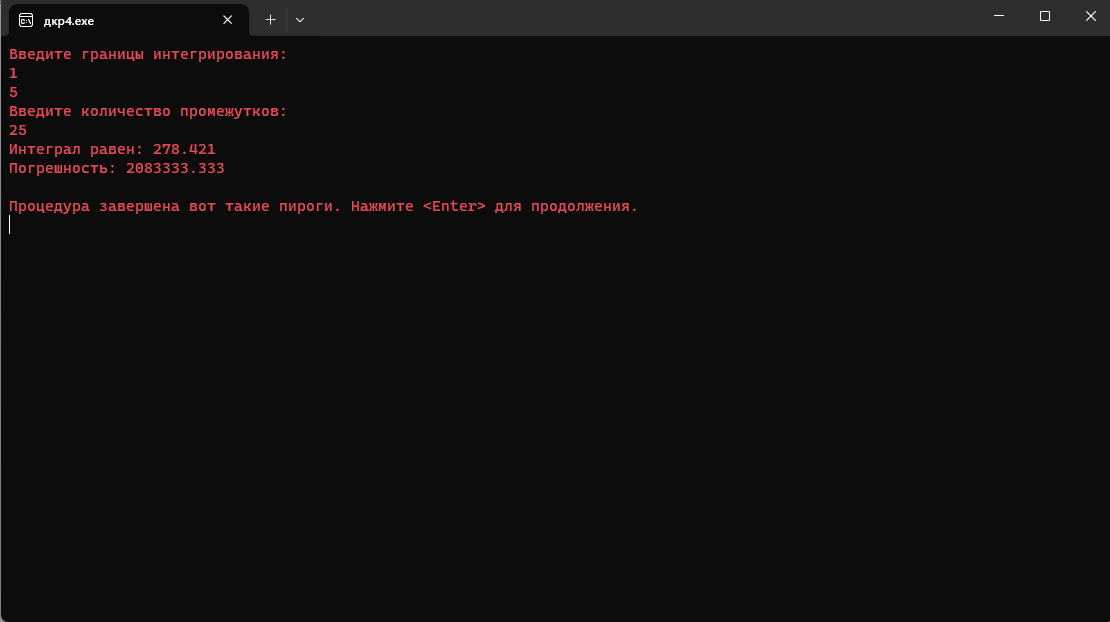


Рис. 8 «Результат интегрирования»

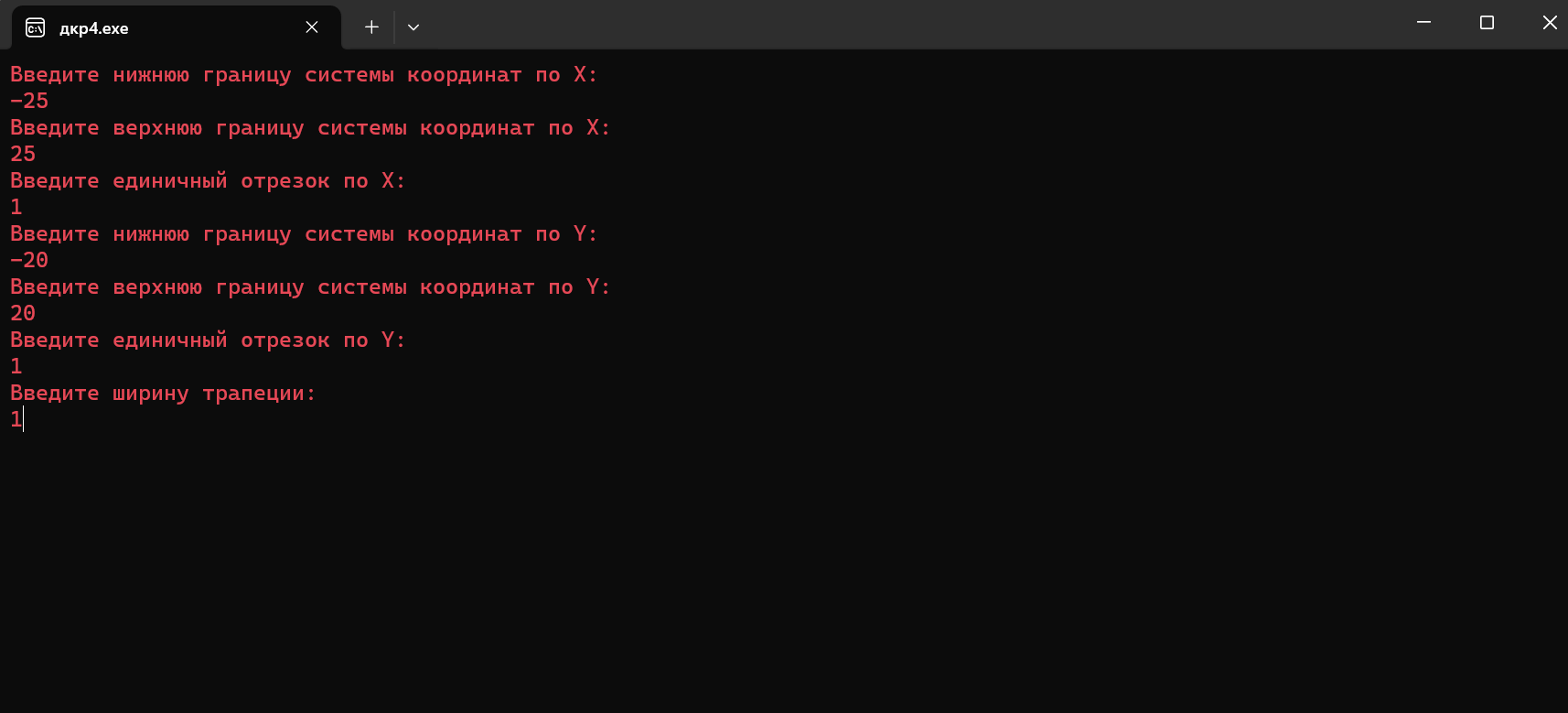


Рис. 9 «Результат интегрирования»

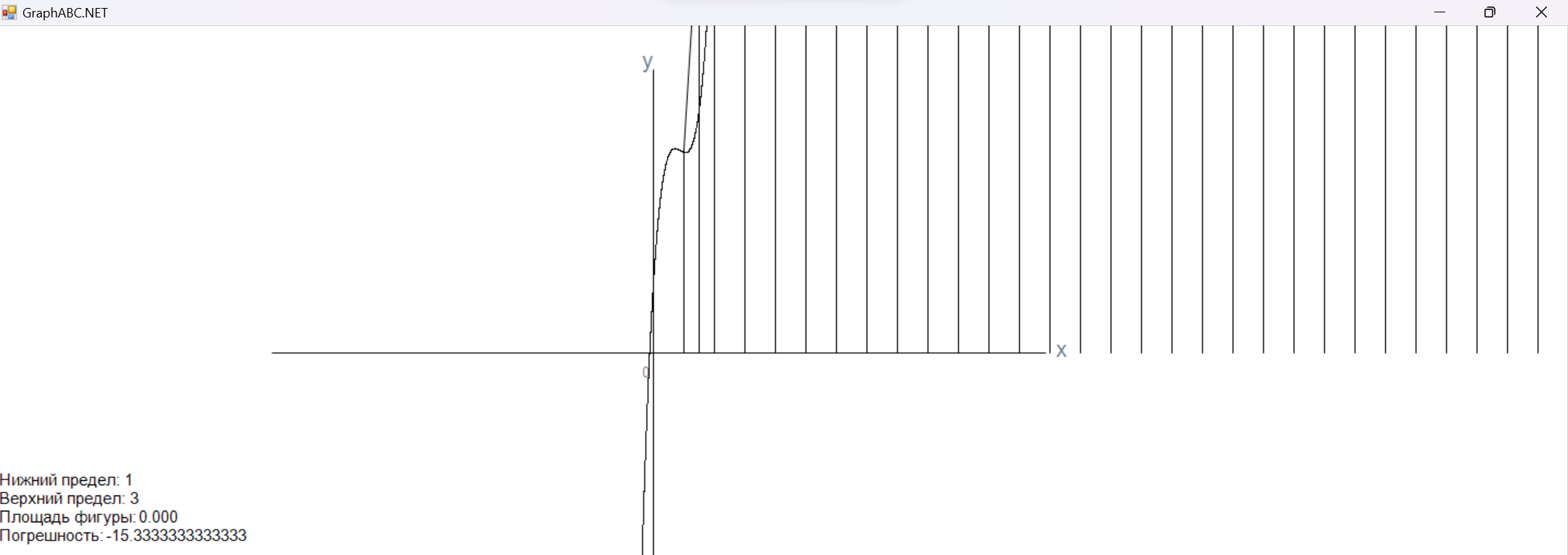


Рис. 10 «Результат интегрирования»

**Вывод:** Лабораторная работа посвящена использованию метода Симпсона для расчета приближенной площади под кривой на заданном интервале и оценки погрешности данного метода. Алгоритм включает в себя определение функции Func(x) для вычисления значений кривой. Пользователю предлагается ввести пределы интегрирования (a, b) и количество промежутков (n), после чего осуществляется вычисление площади фигуры на кривой и оценка погрешности метода Симпсона.

Таким образом, лабораторная работа направлена на применение метода Симпсона для нахождения площади криволинейных фигур, а также на оценку погрешности данного метода, что предоставляет студенту понимание и навыки использования численных методов интегрирования.