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 ПРОГРАМИРОВАНИЯ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Шумилов Иван Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**1.** **Цель работы:**

Цель работы освоить принципы работы в графическом режиме, получить базовые навыки взаимодействия страфическими примитивами.

**2. Формулировка задания:**

**Вариант 24**

Задание состоит из четырех частей:

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.

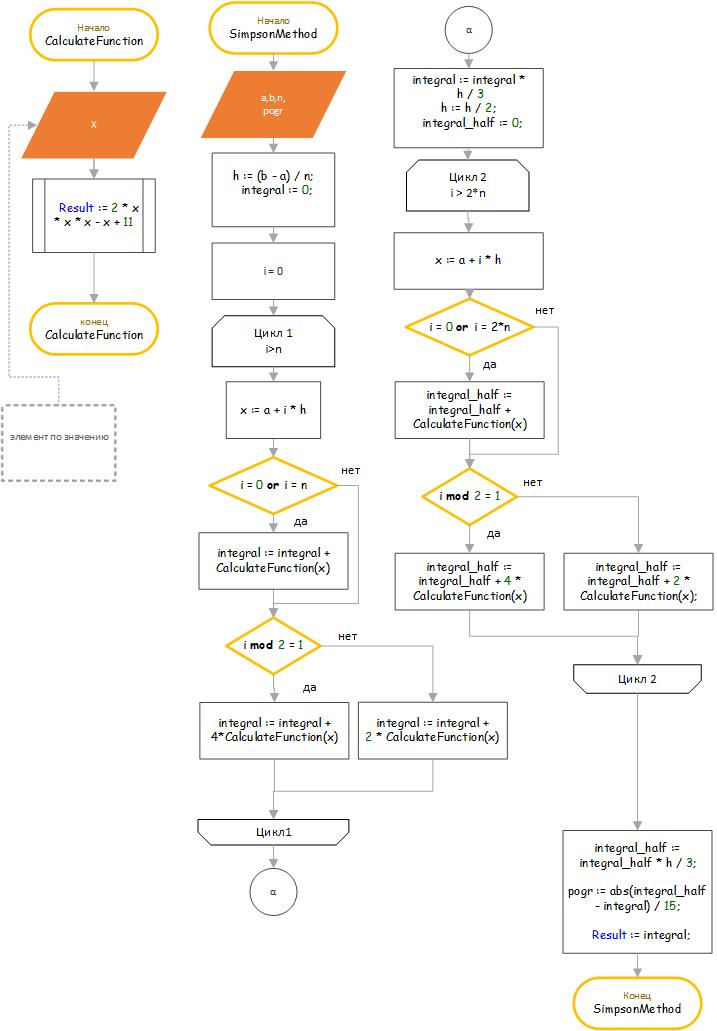
4. Реализовать не менее двух возможностей из представленных независимое масштабирование по осям, штриховка вычисляемой площади, визуализация численного расчета интеграла

**3.1 Описание алгоритма:**

Программа на языке Pascal выполняет две основные задачи: численное интегрирование и визуализацию графика кубической функции.

1. **Численное интегрирование:**
   * Использует метод Симпсона для вычисления приближенной площади под графиком кубической функции в заданных пределах.
   * Пользователь вводит нижний и верхний пределы интегрирования, а также количество делений для уточнения результата.
2. **Визуализация графика:**
   * Использует библиотеку GraphABC для создания графического окна и рисования графика функции.
   * Пользователь задает параметры системы координат, единичные отрезки по осям X и Y.
   * Программа строит график кубической функции, а также выделяет прямоугольники, представляющие фигуру, ограниченную кривой.
   * Выводит значения осей, засечек и другую информацию о графике.
3. **Основная структура программы:**
   * Программа предоставляет пользователю меню с опциями: вычисление площади, построение графика или завершение программы.
   * Выбранное действие выполняется в цикле до тех пор, пока пользователь не выберет завершение программы.

Объединяя численные методы и графическую визуализацию, программа предоставляет пользователю инструмент для анализа функции и оценки площади под ее графиком**.**

**4. Схема алгоритма с комментариями:** 

**Изображение выглядит как диаграмма, текст, План, Технический чертеж

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, диаграмма, Параллельный

Автоматически созданное описание**

**5. Код программы:**

**program** DKR4;

**uses** Crt, GraphABC;

**type**

bam = **function**(x: Real): Real;

**var**

a, b, dx, dy, h: Real;

n: Integer;

s, pogr: Real;

Z: char;

x, y: integer;

color: Byte;

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

**begin**

f := 2 \* (x \*\* 3) + (-2) \* (x \*\* 2) + (-5) \* x + 13;

**end**;

**function** SimpsonMethod(a, b: Real; n: Integer; func: bam; **var** pogr: real): real;

**var**

h, x, integral, integral\_half: real;

i: integer;

**begin**

h := (b - a) / n;

integral := 0;

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

**begin**

x := a + i \* h;

**if** (i = 0) **or** (i = n) **then**

integral := integral + func(x)

**else if** i **mod** 2 = 1 **then**

integral := integral + 4 \* func(x)

**else**

integral := integral + 2 \* func(x);

**end**;

integral := integral \* h / 3;

h := h / 2;

integral\_half := 0;

**for** i := 0 **to** 2 \* n **do**

**begin**

x := a + i \* h;

**if** (i = 0) **or** (i = 2 \* n) **then**

integral\_half := integral\_half + func(x)

**else if** i **mod** 2 = 1 **then**

integral\_half := integral\_half + 4 \* func(x)

**else**

integral\_half := integral\_half + 2 \* func(x);

**end**;

integral\_half := integral\_half \* h / 3;

pogr := abs(integral\_half - integral) / 15;

Result := integral;

**end**;

**procedure** left\_method(a, b: Real; n: Integer; func: bam; **var** s, pogr: real);

**var**

x: Real;

i: Integer;

fa, fb: real; //Значения первообразной

**begin**

h := (b - a) / n;

x := a;

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

**begin**

s := s + func(x);

x := x + h;

**end**;

s := s \* h;

fa := 1 / 2 \* a \*\* 4 - 2 / 3 \* a \*\* 3 - 5 / 2 \* a \*\* 2 + 13 \* a; //нахождение значения первой первообразной через производную

fb := 1 / 2 \* b \*\* 4 - 2 / 3 \* b \*\* 3 - 5 / 2 \* b \*\* 2 + 13 \* b; //второй

pogr := (fb - fa) - s; //нахождение пгрешности

**end**;

**procedure** inf;

**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: ', pogr);

**end**

**else** writeln('Вы не вводили данные');

**end**;

**procedure** graph;

**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(yellow);

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

**repeat**

Z := readkey;

**until** Z = #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); //Размер шрифта

SetFontColor(clSlateGray); //Цвет шрифта

TextOut(xRight + 20, y0 - 15, 'х'); //Подписываем ось OX

TextOut(x0 - 10, yLeft - 30, 'у'); //Подписываем ось OY

SetFontSize(8); //Размер шрифта

SetFontColor(clgray); //Цвет шрифта

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)); //Координата num в окне

Line(x, y0 - 3, x, y0 + 3); //рисуем засечки на оси OX

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); //рисуем засечки на оси Oy

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; //Начальное значение аргумента

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

**begin**

y1 := f(x1); //Вычисляем значение функции

x := x0 + round(x1 \* mx); //Координата Х в графическом окне

y := y0 - round(y1 \* my); //Координата Y в графическом окне

SetPixel(x, y, clred);

x1 := x1 + 0.001 //Увеличиваем абсциссу

**end**;

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

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

setbrushstyle(bsHatch);

setbrushhatch(bhLargeConfetti);

setbrushcolor(clOlive);

x1 := a;

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

**begin**

y1 := f(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);

inf;

**end**;

**begin**

**repeat**

SetConsoleIO;

ClrScr;

textcolor(Yellow);

writeln('1. Вычисление площади фигуры, ограниченной кривой');

writeln('2. График');

writeln('3. Выход');

write('Выберите действие: ');

Z := ReadKey;

**case** Z **of**

'1':

**begin**

ClrScr;

Textcolor(Yellow);

writeln('Введите границы интегрирования: ');

readln(a, b);

Textcolor(Yellow);

writeln('Введите количество делений: ');

readln(n);

s := SimpsonMethod(a, b, n, f, pogr);

Textcolor(LightGreen);

writeln('Площадь фигуры: ', s);

Textcolor(LightGreen);

writeln('Погрешность: ', pogr);

readln;

**end**;

'2': graph;

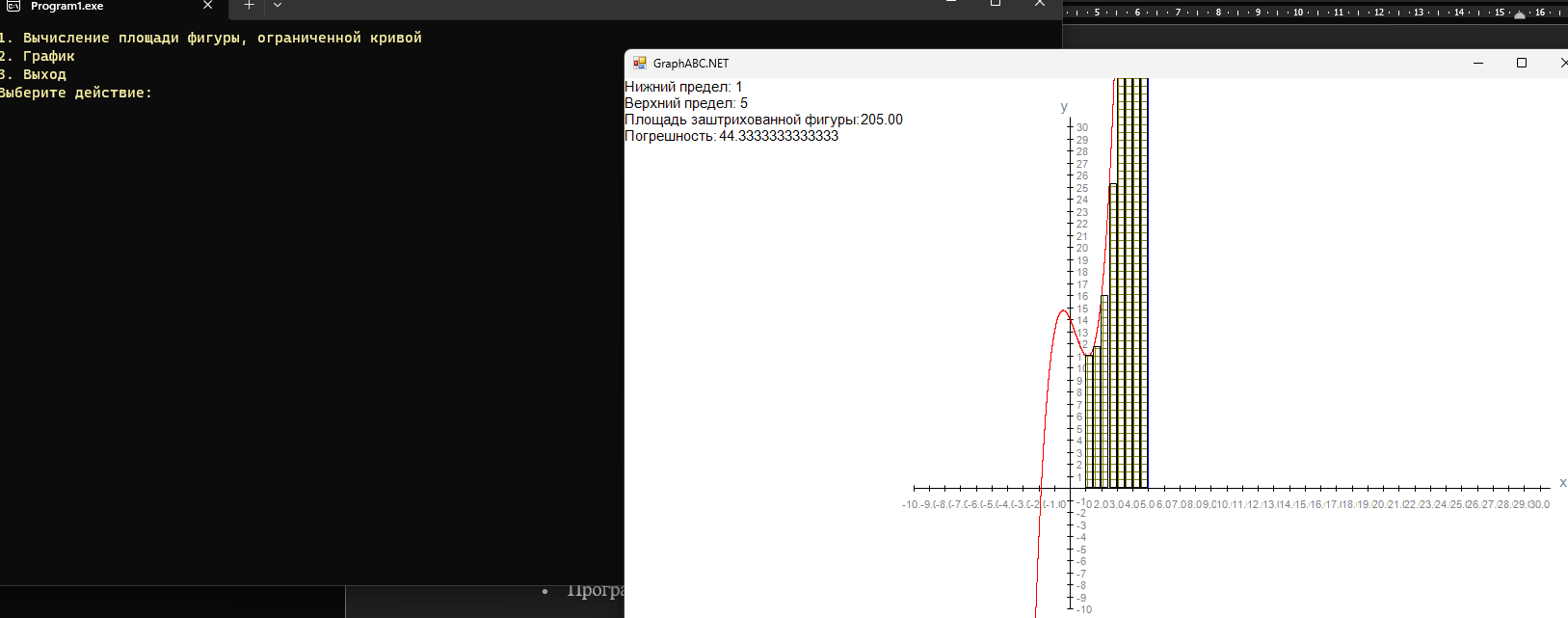
'3': halt;

**end**;

**until** Z = '3';

**end**.

**6.1. Результат выполнения программы**



**7.**  **Вывод:**

В ходе выполнения работы по созданию программы на языке Pascal для численного интегрирования методом Симпсона и визуализации графика функции были достигнуты следующие результаты:

1. **Овладение синтаксисом Pascal:**
   * Успешно изучены принципы создания процедур и функций.
   * Освоены способы передачи данных в подпрограммы.
2. **Разработка программы:**
   * Реализована программа, позволяющая пользователю вводить параметры интегрирования и выбирать между численным интегрированием и построением графика.
   * Программа использует метод Симпсона для численного интегрирования и библиотеку GraphABC для визуализации результатов.
3. **Оценка точности:**
   * Внимание уделено оценке погрешности численного метода с использованием простого сравнения результатов на разных шагах.
4. **Графический интерфейс:**
   * Разработан минимальный, но функциональный графический интерфейс, позволяющий пользователю визуально оценить график функции и результаты численного интегрирования.
5. **Информативный вывод:**
   * Программа предоставляет пользователю информативный вывод, включающий параметры интегрирования, площадь фигуры и оценку погрешности.

Объединение численных методов, графической визуализации и интерфейса пользователя позволяет программе быть гибким инструментом для решения задач анализа функций. Полученные навыки работы с языком Pascal могут быть применены при разработке более сложных программ в будущем.

Начало формы