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»**

Выполнил: студент учебной группы

ИСПк-101-51-00

Клыков Денис Борисович

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

В отчете должны отображаться:

1. Цель работы
2. Формулировка задания (с вариантом)
3. Описание алгоритма и ответы на вопросы
4. Схема алгоритма с комментариями
5. Код программы
6. Результат выполнения программы
7. Вывод

Цель работы: Освоить принципы работы в графическом режиме, получить базовые навыки взаимодействия примитивами.

Задание

1. Дополнить программу, реализованную в ходе предыдущей лабораторной работы, режимом визуализации.  
2. Предусмотреть возможность вывода кривой, ограниченной фигуру, на координатную плоскость.  
3. Реализовать следующие возможности и элементы: масштабные графика, подписки 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.  
**program** AreaUnderCurveVisualization;

**uses**

GraphABC;

**type**

TFunction = **function**(x: Double): Double;

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

**begin**

f := 2 \* x \* x \* x + 2 \* x \* x - 2 \* x + 13;

**end**;

**function** LeftRectangles(a, b: Double; n: Integer; func: TFunction): Double;

**var**

h, x: Double;

i: Integer;

**begin**

h := (b - a) / n;

Result := 0;

x := a;

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

**begin**

Result := Result + func(x);

x := x + h;

**end**;

Result := Result \* h;

**end**;

**function** EstimateError(a, b: Double; n: Integer; func: TFunction): Double;

**var**

integral1, integral2: Double;

**begin**

integral1 := LeftRectangles(a, b, n, func);

integral2 := LeftRectangles(a, b, 2 \* n, func);

Result := Abs(integral2 - integral1);

**end**;

**procedure** DrawAxes;

**begin**

Line(0, WindowHeight **div** 2, WindowWidth, WindowHeight **div** 2); { Ось X }

Line(WindowWidth **div** 2, 0, WindowWidth **div** 2, WindowHeight); { Ось Y }

**end**;

**procedure** DrawFunction(func: TFunction);

**var**

xScreen, yScreen: integer;

xReal: real;

**begin**

**for** xScreen := 0 **to** WindowWidth **do**

**begin**

xReal := (xScreen - WindowWidth **div** 2) / 20; { Масштабирование по оси X }

yScreen := WindowHeight **div** 2 - Round(func(xReal) \* 20); { Масштабирование по оси Y }

**if** (yScreen >= 0) **and** (yScreen <= WindowHeight) **then**

SetPixel(xScreen, yScreen, clWhite);

**end**;

**end**;

**procedure** DrawInfo;

**begin**

SetBrushColor(clBlack);

SetFontColor(clWhite);

TextOut(10, 10, 'Function: f(x) = 2\*x^3 + 2\*x^2 - 2\*x + 13');

TextOut(10, 30, 'X-axis: scaled by factor of 20');

TextOut(10, 50, 'Y-axis: scaled by factor of 20');

**end**;

**procedure** ShadingArea(a, b: real; n: integer; func: TFunction);

**var**

xScreen, yScreen: integer;

xReal, h: real;

**begin**

h := (b - a) / n;

**for** xScreen := Round(a \* 20 + WindowWidth **div** 2) **to** Round(b \* 20 + WindowWidth **div** 2) **do**

**begin**

xReal := (xScreen - WindowWidth **div** 2) / 20;

yScreen := WindowHeight **div** 2 - Round(func(xReal) \* 20);

Line(xScreen, WindowHeight **div** 2, xScreen, yScreen);

**end**;

**end**;

**procedure** VisualizeIntegralCalculation(a, b: real; n: integer; func: TFunction);

**var**

i, xScreen: integer;

xReal, h: real;

**begin**

h := (b - a) / n;

SetBrushColor(clRed);

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

**begin**

xReal := a + i\*h;

xScreen := Round((xReal - a) \* 20) + WindowWidth **div** 2;

Rectangle(xScreen, WindowHeight **div** 2, xScreen + Round(h \* 20), WindowHeight **div** 2 - Round(func(xReal) \* 20));

**end**;

SetBrushColor(clBlack);

**end**;

**procedure** CalculateArea;

**var**

a, b: Double;

n: Integer;

area, error: Double;

**begin**

Write('Введите нижний предел интегрирования (a): ');

ReadLn(a);

Write('Введите верхний предел интегрирования (b): ');

ReadLn(b);

Write('Введите количество разбиений (n): ');

ReadLn(n);

area := LeftRectangles(a, b, n, f);

error := EstimateError(a, b, n, f);

WriteLn('Площадь фигуры: ', area:0:6);

WriteLn('Оценка погрешности: ', error:0:6);

SetWindowWidth(800);

SetWindowHeight(600);

ClearWindow(clBlack);

DrawAxes;

DrawFunction(f);

DrawInfo;

ShadingArea(a, b, n, f);

VisualizeIntegralCalculation(a, b, n, f);

**end**;

**procedure** DisplayMenu;

**begin**

WriteLn('1. Вычислить площадь фигуры');

WriteLn('2. Визуализировать график');

WriteLn('3. Выход');

Write('Выберите действие: ');

**end**;

**var**

choice: Integer;

**begin**

**repeat**

DisplayMenu;

ReadLn(choice);

**case** choice **of**

1: CalculateArea;

2: CalculateArea; { В данном случае, расчет площади включает визуализацию }

3: WriteLn('Выход из программы...');

**else**

WriteLn('Неверный выбор. Попробуйте снова.');

**end**;

WriteLn;

**until** choice = 3;

**end**.