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-205-52-00 Головёшкин Андрей Евгеньевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2024

**Вариант 2**

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

**Задание:**

1. Дополнить программу, реализованную в ходе предыдущей лабораторной работы, режимом визуализации.

2. Предусмотреть возможность вывода кривой, ограничивающей фигуру, на координатную

плоскость.

**Описание алгоритма:**

1. **Инициализация окна:**
   * Устанавливается размер окна графического интерфейса: ширина — 800 пикселей, высота — 600 пикселей.
   * Устанавливается цвет пера для рисования линий: черный (clBlack).
   * Устанавливается толщина пера для рисования линий: 2 пикселя.
2. **Рисование осей координат:**
   * Рисуется горизонтальная ось X от точки (50, 300) до точки (750, 300).
   * Рисуется вертикальная ось Y от точки (400, 50) до точки (400, 550).
3. **Масштабирование графика:**
   * Рассчитывается масштаб для оси X: scaleX = 600 / (b - a), где b - a — это длина интервала на оси X.
   * Рассчитывается масштаб для оси Y: scaleY = 250 / (f(b) - f(a)), где f(b) - f(a) — это диапазон значений функции на интервале от a до b.
4. **Построение графика функции:**
   * Переменная x инициализируется значением a.
   * Для каждой точки функции на интервале от a до b (с шагом 0.01) выполняются следующие действия:
   * Вычисляется значение функции y = f(x).
   * Преобразуется координата x в экранную координату screenX с учетом масштаба: screenX = Round(50 + (x - a) \* scaleX).
   * Преобразуется координата y в экранную координату screenY с учетом масштаба: screenY = Round(300 - (y - f(a)) \* scaleY).
   * Рисуется линия от предыдущей точки до текущей, образуя график функции.
5. **Заполнение графика под графиком функции:**
   * Устанавливается цвет заливки: синий (clBlue).
   * Для каждой точки функции на интервале от a до b (с шагом 0.01) выполняются следующие действия:
   * Вычисляется значение функции y = f(x).
   * Преобразуется координата x в экранную координату screenX.
   * Преобразуется координата y в экранную координату screenY.
   * Если screenY не превышает 300 (то есть точка находится ниже оси X), рисуется прямоугольник от уровня оси X до функции, заполняя область под графиком синим цветом.
6. **Завершение:**
   * График рисуется в окне, и процедура завершает выполнение.
   * Визуальный эффект:
   * График функции отображается как линия, соединяющая множество точек, каждая из которых соответствует значению функции на заданном интервале.
   * Под графиком функции (если она выше оси X) заполняется область синим цветом, представляя собой интеграл функции на заданном интервале.

**Схема алгоритма**

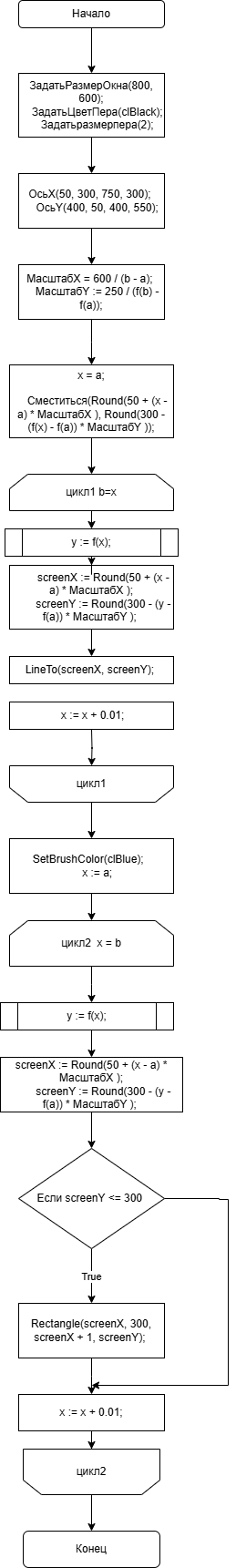


Схема задание

Схема задание

**Коды программ**

**1.**

uses

Crt, GraphABC;

function f(x: real): real;

begin

f := Power(x, 3) + Power(x, 2) - x + 1;

end;

function MidpointRectangleMethod(a, b: real; n: integer): real;

var

h, mid\_point, area: real;

i: integer;

begin

h := (b - a) / n;

area := 0;

for i := 0 to n - 1 do

begin

mid\_point := a + (i + 0.5) \* h;

area := area + f(mid\_point) \* h;

end;

MidpointRectangleMethod := area;

end;

function EstimateError(a, b: real; n: integer): real;

var

h, area\_with\_trapezoids: real;

i: integer;

begin

h := (b - a) / n;

area\_with\_trapezoids := (f(a) + f(b)) \* (b - a) / 2;

for i := 1 to n - 1 do

begin

area\_with\_trapezoids := area\_with\_trapezoids + f(a + i \* h) \* h;

end;

EstimateError := Abs(area\_with\_trapezoids - MidpointRectangleMethod(a, b, n));

end;

procedure DrawGraph(a, b: real);

var

x, y: real;

screenX, screenY: integer;

scaleX, scaleY: real;

begin

SetWindowSize(800, 600);

SetPenColor(clBlack);

SetPenWidth(2);

Line(50, 300, 750, 300);

Line(400, 50, 400, 550);

scaleX := 600 / (b - a);

scaleY := 250 / (f(b) - f(a));

x := a;

MoveTo(Round(50 + (x - a) \* scaleX), Round(300 - (f(x) - f(a)) \* scaleY));

while x <= b do

begin

y := f(x);

screenX := Round(50 + (x - a) \* scaleX);

screenY := Round(300 - (y - f(a)) \* scaleY);

LineTo(screenX, screenY);

x := x + 0.01;

end;

SetBrushColor(clBlue);

x := a;

while x <= b do

begin

y := f(x);

screenX := Round(50 + (x - a) \* scaleX);

screenY := Round(300 - (y - f(a)) \* scaleY);

if screenY <= 300 then

Rectangle(screenX, 300, screenX + 1, screenY);

x := x + 0.01;

end;

end;

var

a, b: real;

n: integer;

area, error: real;

begin

writeln('Программа для вычисления площади фигуры, ограниченной кривой и осью X.');

repeat

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

readln(a);

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

readln(b);

if (a < 0) or (b < 0) or (a >= b) then

writeln('Ошибка: входные значения должны быть положительными и a < b.');

until (a >= 0) and (b >= 0) and (a < b);

repeat

write('Введите число подынтервалов (n): ');

readln(n);

if n <= 0 then

writeln('Ошибка: число подынтервалов должно быть положительным.');

until n > 0;

area := MidpointRectangleMethod(a, b, n);

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

writeln;

writeln('Вычисленная площадь: ', area:0:6);

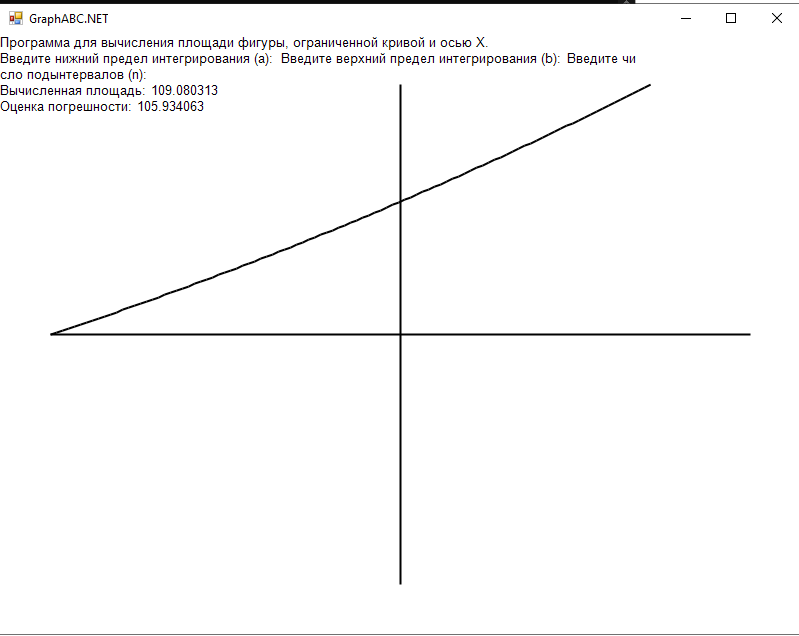
writeln('Оценка погрешности: ', error:0:6);

DrawGraph(a, b);

end.

**Результат выполнения программы**

**1.**

****

**Вывод:**

Программа выполняет численное вычисление площади, ограниченной графиком функции и осью xxx, на заданном интервале. Она использует метод средних прямоугольников для приближенного вычисления площади и метод трапеций для оценки погрешности. Пользователь вводит пределы интегрирования и количество подынтервалов, после чего программа выводит результат вычислений и оценку погрешности. Также строится график функции, что помогает визуализировать процесс интегрирования.