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 ПРОГРАММИРИВАНИЯ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Балахнина Елизавета Ефимовна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

Содержание

1. Цель лабораторной работы
2. Формулировка задания (с вариантом)
3. Описание алгоритма
4. Схема алгоритма с комментариями
5. Код программы
6. Результат выполнения программы
7. Вывод
8. **Цель работы:** освоить принципы работы в графическом режиме, получить базовые навыки взаимодействия с графическими примитивами.
9. **Формулировка задания (Вариант:2)**

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.

1. **Описание алгоритма**

Данный алгоритм реализует метод средних прямоугольников для приближенного вычисления определенного интеграла функции.

1. Первым шагом предлагается ввести нижний и верхний пределы интегрирования, а также количество прямоугольников для разбиения области интегрирования.

2. Задаются начальные параметры для отрисовки графика функции.

3. Отрисовывается график функции в заданных пределах с помощью пикселей на экране.

4. Рассчитывается ширина каждого прямоугольника (h) на основе количества прямоугольников и заданных пределов.

5. Инициализируется переменная суммы (sum) для сохранения результата.

6. Начиная с нижнего предела интегрирования (a), цикл проходит через каждый прямоугольник.

7. Для каждого прямоугольника, сумма увеличивается на произведение значения функции в средней точке прямоугольника и его ширины (функция \* h).

8. После завершения цикла, выводится значение интеграла, рассчитанного с помощью метода средних прямоугольников (sum).

1. **Схема алгоритма с комментариями**

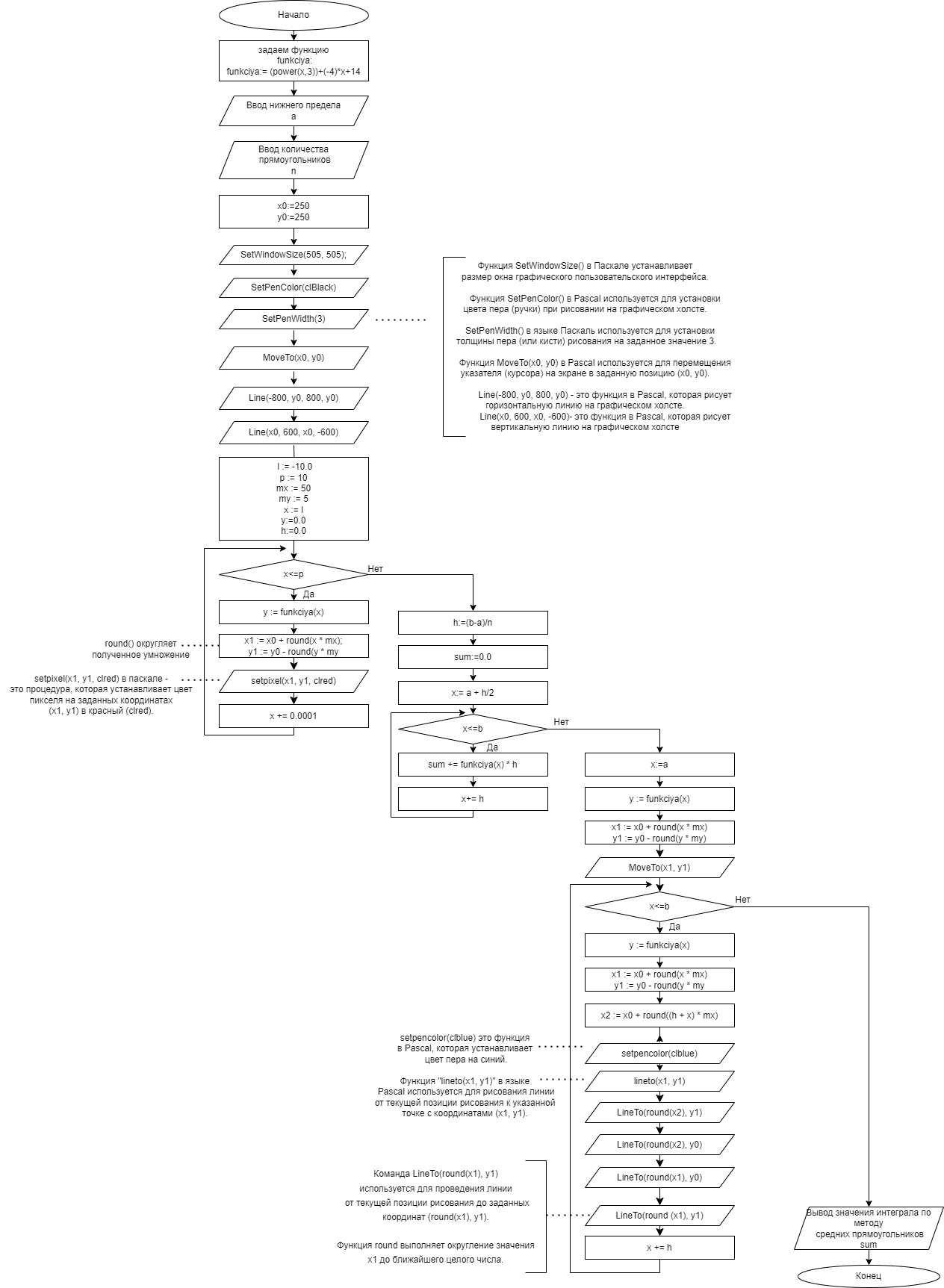


Рис 1. Схема алгоритма с комментариями.

1. **Код программы**

**uses** crt;

**uses** graphABC;

**var**

mx, my, e,p: real;

a, b, x0, y0, n, w, l, k: integer;

casem, vibo: byte;

**function** F1(x: real): real;

**begin**

F1 := 1 \* power(x, 3) + 1 \* power(x, 2) + (4) \* x + 6;

**end**;

**function** vibor(**var** s: byte): byte;

**begin**

Textcolor(2);

writeln('Увеличить по x - 1');

Textcolor(4);

writeln('Увеличить по y - 2');

Textcolor(15);

writeln('Увеличить по x и y - 3');

Writeln('Выберите вариант: ');

readln(s);

vibor := s;

**end**;

**function** grapx(**var** s1, s2, h: real; ras: integer): integer;

**var**

i: integer; x: real;

**begin**

ClearWindow;

a := -5;

b := 50;

x0 := windowwidth **div** 2;

y0 := windowheight **div** 2;

vibo := vibor(vibo);

**case** vibo **of**

1:

**begin**

writeln('Введите размер промежутка по x');

readln(mx);

my := 10;

**end**;

2:

**begin**

writeln('Введите размер промежутка по y');

mx := 10;

readln(my);

**end**;

3:

**begin**

writeln('Введите размер промежутка по y');

readln(my);

writeln('Введите размер промежутка по x');

readln(mx);

**end**;

**end**;

line(0, y0, windowwidth, y0);

line(x0, 0, x0, windowheight);

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

**begin**

line(x0 + round(i \* mx), y0 - 3, x0 + round(i \* mx), y0 + 3);

line(x0 - round(i \* mx), y0 - 3, x0 - round(i \* mx), y0 + 3);

textout(x0 + round(i \* mx), y0 + 10, inttostr(i));

textout(x0 - round(i \* mx), y0 + 5, inttostr(-i));

**end**;

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

**begin**

line(x0 - 3, y0 + round(i \* my), x0 + 3, y0 + round(i \* my));

line(x0 - 3, y0 - round(i \* my), x0 + 3, y0 - round(i \* my));

textout(x0 - 25, y0 - round(i \* my), inttostr(i));

textout(x0 - 20, y0 + round(i \* my), inttostr(-i));

**end**;

textout(x0 + 5, y0 + 10, '0');

textout(windowwidth - 10, y0 - 15, 'X');

textout(x0 + 5, 10, 'Y');

line(x0 + round(s1 \* mx), 0, x0 + round(s1 \* mx), windowheight);

line(x0 + round(s2 \* mx), 0, x0 + round(s2 \* mx), windowheight);

l := round((x0 + round(s2 \* mx) - x0 - round(s1 \* mx)) / ras);

k := l;

p :=(s2-s1)/ras;

w := 0;

x := a;

**while** x <= b **do**

**begin**

setpixel(x0 + round(x \* mx), y0 - round(F1(x) \* my), clred);

**if** (x0 + round(s1 \* mx)) = (x0 + round(x \* mx)) **then**

**begin**

setpencolor(clnavy);

e := x;

Rectangle(x0 + round(s1 \* mx), round(y0 - round(F1(p/2+e)\*my)), x0 + round(s1 \* mx + (k + l)), y0);

**end**;

**if** (x0 + round(s1 \* mx + l)) = (x0 + round(x \* mx)) **then**

**begin**

setpencolor(clnavy);

e := x;

**for var** v := x0 + round(s1 \* mx) **to** x0 + round(s2 \* mx) **do**

Rectangle(x0 + round(s1 \* mx + l), round(y0 - round(F1(p/2+e) \* my) ), x0 + round(s1 \* mx + (k + l)), y0);

l := l + k;

e := e + l;

inc(w);

**if** (w + 1) = ras **then**

**begin**

setpencolor(clblack);

**break**;

**end**;

**end**;

x := x + 0.001;

**end**;

textout(10, 10, 'Найти площади фигуры, ограниченной кривой 1\*x^3+(1)\*x^2+(4)\*x+(6) и осью Ох (в положительной части по оси Оу)');

textout(10, 900, 'Нажмите на "Enter" что бы продолжить');

readln;

ClearWindow;

**end**;

**function** grap(s1, s2, h: real; ras: integer): integer;

**var**

i: integer; x: real;

**begin**

ClearWindow;

a := -5;

b := 50;

x0 := windowwidth **div** 2;

y0 := windowheight **div** 2;

mx := 10;

my := 10;

line(0, y0, windowwidth, y0);

line(x0, 0, x0, windowheight);

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

**begin**

line(x0 + round(i \* mx), y0 - 3, x0 + round(i \* mx), y0 + 3);

line(x0 - round(i \* mx), y0 - 3, x0 - round(i \* mx), y0 + 3);

textout(x0 + round(i \* mx), y0 + 10, inttostr(i));

textout(x0 - round(i \* mx), y0 + 5, inttostr(-i));

**end**;

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

**begin**

line(x0 - 3, y0 + round(i \* my), x0 + 3, y0 + round(i \* my));

line(x0 - 3, y0 - round(i \* my), x0 + 3, y0 - round(i \* my));

textout(x0 - 25, y0 - round(i \* my), inttostr(i));

textout(x0 - 20, y0 + round(i \* my), inttostr(-i));

**end**;

textout(x0 + 5, y0 + 10, '0');

textout(windowwidth - 10, y0 - 15, 'X');

textout(x0 + 5, 10, 'Y');

line(x0 + round(s1 \* mx), 0, x0 + round(s1 \* mx), windowheight);

line(x0 + round(s2 \* mx), 0, x0 + round(s2 \* mx), windowheight);

l := round((x0 + round(s2 \* mx) - x0 - round(s1 \* mx)) / ras);

k := l;

p:=(s2-s1)/ras;

w := 0;

x := a;

**while** x <= b **do**

**begin**

setpixel(x0 + round(x \* mx), y0 - round(F1(x) \* my), clred);

**if** (x0 + round(s1 \* mx)) = (x0 + round(x \* mx)) **then**

**begin**

setpencolor(clnavy);

e := x;

Rectangle(x0 + round(s1 \* mx), round(y0 - round(F1(p/2+e) \* my) \* 1.4), x0 + round(s1 \* mx + (k + l)), y0);

**end**;

**if** (x0 + round(s1 \* mx + l)) = (x0 + round(x \* mx)) **then**

**begin**

setpencolor(clnavy);

e := x;

**for var** v := x0 + round(s1 \* mx) **to** x0 + round(s2 \* mx) **do**

Rectangle(x0 + round(s1 \* mx + l), round(y0 - round(F1(p/2+e) \* my) \* 1.4), x0 + round(s1 \* mx + (k + l)), y0);

l := l + k;

e := e + l;

inc(w);

**if** (w + 1) = ras **then**

**begin**

setpencolor(clblack);

**break**;

**end**;

**end**;

x := x + 0.001;

**end**;

textout(10, 10, 'Найти площади фигуры, ограниченной кривой 1\*x^3+(1)\*x^2+(4)\*x+(6) и осью Ох (в положительной части по оси Оу)');

textout(10, 600, 'Нажмите на "Enter" что бы продолжить');

readln;

ClearWindow;

**end**;

**function** casemf(**var** v: byte): byte;

**begin**

Textcolor(2);

writeln('Да - 1');

Textcolor(4);

writeln('Нет - 0');

Textcolor(15);

Writeln('Выберите вариант: ');

readln(v);

casemf := v;

**end**;

**function** func(**var** x: real): real;

**begin**

**var** f1: real;

f1 := 1 \* power(x, 3) + 1 \* power(x, 2) + (4) \* x + 6;

func := f1;

**end**;

**function** func1(**var** x: real): real;

**begin**

**var** f2: real;

f2 := 1 / 4 \* power(x, 4) + 1 / 3 \* power(x, 3) + 2 \* power(x, 2) + 6 \* x;

func1 := f2;

**end**;

**function** predel: integer;

**begin**

ClrScr;

**var** a, b, h, f, x, S: real;

**var** n: integer;

Textcolor(5);

Writeln('Вычисление площади фигуры, ограниченной кривой 1\*x^3+(1)\*x^2+(4)\*x+(6) и осью Ох (в положительной части по оси Оу)');

Textcolor(3);

print('Введите пределы интегрирования "a"<"b":');

read(a, b);

writeln(a, ' ', b);

print('Количество интервалов разбиения:');

read(n);

writeln(n);

h := (b - a) / n;

x := a + h;

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

**begin**

f := func(x);

S := S + f;

x := x + h;

**end**;

S := (h \* (func(a) + func(b))) / 2 + S;

Textcolor(10);

writeln('Ответ ', S);

writeln;

Textcolor(15);

writeln('Вывести погрешность полученного результата?');

casem := casemf(casem);

**case** casem **of**

1:

**begin**

Textcolor(6);

writeln('Погрешность = ', abs((func1(b) - func1(a)) - S));

**end**;

0:

**else**

textcolor(black);

Textbackground(red);

writeln('Нет такого выбора!!!');

textbackground(black);

textcolor(white);

**end**;

writeln;

writeln('Нарисовать график?');

casem := casemf(casem);

**case** casem **of**

1: grap(a, b, h, n);

0:

**else**

textcolor(black);

Textbackground(red);

writeln('Нет такого выбора!!!');

textbackground(black);

textcolor(white);

**end**;

writeln('Может увеличить размер графика?');

casem := casemf(casem);

**case** casem **of**

1: grapx(a, b, h, n);

0:

**else**

textcolor(black);

Textbackground(red);

writeln('Нет такого выбора!!!');

textbackground(black);

textcolor(white);

**end**;

writeln;

Textcolor(15);

writeln('Начать заново?');

casem := casemf(casem);

**case** casem **of**

1: predel;

0:

**begin**

textcolor(red);

writeln('Спасибо за использование данной программы, ждём снова!');

textcolor(white);

**exit**()

**end**

**else**

textcolor(black);

Textbackground(red);

writeln('Нет такого выбора!!!');

writeln('Перезапустите программу!!!');

textbackground(black);

textcolor(white);

**end**;

predel := 0;

**end**;

**begin**

ClrScr;

MaximizeWindow;

Textcolor(5);

Writeln('Вычисление площади фигуры, ограниченной кривой 1\*x^3+(1)\*x^2+(4)\*x+(6) и осью Ох (в положительной части по оси Оу)');

Textcolor(3);

Writeln('Ввести пределы интегрирования в ручную?');

casem := casemf(casem);

**case** casem **of**

1: predel;

0:

**begin**

textcolor(red);

writeln('Спасибо за использование данной программы, ждём снова!');

textcolor(white);

**exit**()

**end**

**else**

textcolor(black);

Textbackground(red);

writeln('Нет такого выбора!!!');

writeln('Перезапустите программу!!!');

textbackground(black);

textcolor(white);

**end**;

**end**.

1. **Результат выполнения программы**

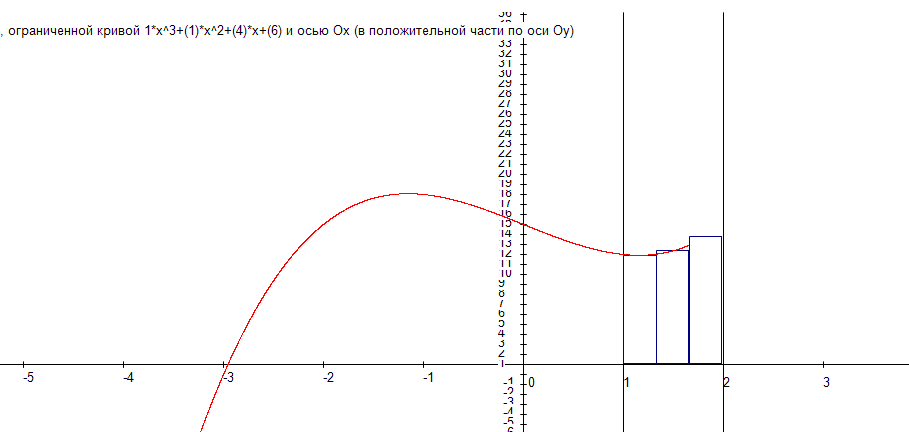
******

Рис.2. Результат выполнения программы.

**Вывод**

В процессе выполнения домашней контрольной работы, мы узнали множество новых операций на языке Паскаль, как например, модуль GraphABC.

В языке программирования Паскаль (Pascal), GraphABC — это модуль программирования, который предоставляет пользователю простой интерфейс для создания и отображения графических объектов, таких как точки, линии, фигуры и т.д. Этот модуль позволяет пользователю создавать и изменять графики, используя программный код, без необходимости знания сложных математических формул или графических алгоритмов. Модуль GraphABC облегчает процесс создания визуальных элементов и позволяет пользователям визуализировать данные и создавать графические приложения.

К тому же, мы узнали множество процедур и функций. Некоторые из них:

1. Функция SetWindowSize() в Паскале устанавливает размер окна графического пользовательского интерфейса.
2. Функция SetPenColor() в Pascal используется для установки цвета пера (ручки) при рисовании на графическом холсте.
3. SetPenWidth() в языке Паскаль используется для установки толщины пера (или кисти) рисования на заданное значение 3.
4. Функция MoveTo (x0, y0) в Pascal используется для перемещения указателя (курсора) на экране в заданную позицию (x0, y0).
5. Line() — это функция в Pascal, которая рисует горизонтальную линию на графическом холсте.
6. Line() — это функция в Pascal, которая рисует вертикальную линию на графическом холсте.

Также в процессе выполнения работы, мы познакомились с сайтом diagrams.net. Diagrams.net - это бесплатный онлайн-инструмент для создания диаграмм, схем, графиков и других визуальных элементов. Он представляет удобный интерфейс с возможностью создания различных типов диаграмм, таких как блок-схемы, организационные системы и многие другие.