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

Ильин Тимофей Анатольевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров 2023

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

**Задание 1 (Вариант 7)**

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.

**Описание алгоритма**

1. Процедура casemenu принимает на вход два вещественных числа a и b, которые задают границы отрезка, на котором вычисляется интеграл, а также две переменные area и error, которые будут использоваться для хранения значения интеграла и погрешности соответственно.
2. Внутри процедуры определяются и инициализируются необходимые переменные: n - количество разбиений отрезка, i - счетчик для цикла, x и h - вспомогательные переменные для вычисления значений функции f(x) на различных точках отрезка, sum1 и sum2 - промежуточные суммы.
3. Задается значение переменной n равное 100, что означает, что отрезок [a, b] будет разбит на 100 равных частей.
4. Вычисляется величина h, которая представляет собой длину каждого подотрезка и равна (b - a) / n.
5. Инициализируются переменные sum1 и sum2 нулевыми значениями.
6. В цикле от 1 до n div 2 (целая часть от деления n на 2) производятся вычисления значения функции f(x) в двух точках каждого подотрезка и добавляются к промежуточным суммам sum1 и sum2 соответственно.
7. После выполнения цикла получаем значения сумм sum1 и sum2, которые соответствуют треугольным частям площадей на графике функции f(x).
8. Вычисляется значение интеграла с использованием параболического правила Симпсона: sum = h / 3 \* (f(a) + 4 \* sum1 + 2 \* sum2 + f(b)), где f(a) и f(b) - значения функции на границах отрезка.
9. Вычисляется значение площади под графиком функции (интеграла) - area, как абсолютное значение значения sum.
10. Вычисляется значение погрешности вычисления интеграла - error, как абсолютное значение разности значений sum и area.
11. Результаты вычислений интеграла и погрешности сохраняются в переменных area и error соответственно.
12. Объявляем переменные a, b, area, error как вещественные числа и choice как целое число.
13. В начале процедуры draw\_plane задаются начальные значения переменных x и y, которые указывают на начальные точки по осям X и Y соответственно.
14. Затем задается размер окна, название окна и цвет пера (черный).
15. Далее начинается отрисовка осей X и Y, а также отметок на осях.
16. Затем выполняется цикл от 1 до b, в котором отрисовываются вертикальные линии и текстовые отметки на оси X.
17. В следующем цикле от 1 до 36 отрисовываются горизонтальные линии и текстовые отметки на оси Y.
18. Таким образом, процедура draw\_plane отрисовывает плоскость координат с заданными масштабом и значениями по осям X и Y.
19. Внутри функции определены следующие переменные:

i: переменная для цикла;

h: шаг на отрезке.

1. Первое действие в функции - установка переменной n в значение 10.
2. Далее, переменной h присваивается значение (b - a) / n, что позволяет определить шаг между точками графика.
3. Затем, идет цикл for, который выполняется от 0 до n - 1.
4. Внутри цикла происходит следующее:
5. Устанавливается цвет пера на красный и устанавливается ширина пера на 2 пикселя.
6. Отрисовывается линия между двумя точками. Координаты точек рассчитываются с помощью следующей формулы:

x1 = trunc(x + a \* 20 \* scale + h \* i \* 20 \* scale)

y1 = y - trunc(F(a + i \* h)) \* 20 \* scale

x2 = trunc(x + a \* 20 \* scale + h \* i \* 20 \* scale + h \* 20 \* scale)

y2 = y - trunc(F(a + i \* h + h)) \* 20 \* scale

1. Цикл повторяется для всех точек графика.
2. В конце функции возращается значение типа boolean, которое в данном случае не используется.
3. Функция rects начинается с определения переменной n, которая устанавливается равной 100.
4. Далее вычисляется значение переменной h, которая представляет собой разницу между b и a, деленную на n.
5. Затем происходит цикл с итерацией от 0 до n-1 (т.е. от 0 до 99). В каждой итерации цикла происходит отрисовка прямоугольника с использованием функций SetBrushColor и rectangle.
6. Функция SetBrushColor устанавливает цвет заливки прямоугольника в цвет clBlue (синий).
7. Функция rectangle отрисовывает прямоугольник с заданными координатами, включая начальную и конечную точку, используя функцию trunc для округления рассчитанных значений.
8. Ширина прямоугольника рассчитывается с использованием формулы: a \* 20 \* scale + h \* i \* 20 \* scale + h \* 20 \* scale.
9. Высота прямоугольника рассчитывается с использованием функции F, которая принимает в качестве аргумента значение a + i \* h + h - h / 2 и округляет его с помощью функции trunc. Затем это значение умножается на 20 \* scale.
10. В конце функция rects возвращает значение типа boolean, которое указывает, успешно ли была выполнена отрисовка прямоугольников.
11. Начинаем цикл repeat...until, который будет выполняться до выбора пользователем пункта меню "Выход" (choice = 2).
12. Выводим на экран пункты меню: "Вычислить площадь фигуры" и "Выход".
13. Просим пользователя выбрать пункт меню и считываем его выбор в переменную choice.
14. Проверяем выбор пользователя с помощью оператора case. Если пользователь выбрал пункт 1, выполняем следующий набор действий:
15. Просим пользователя ввести нижнюю границу интегрирования и считываем ее в переменную a.
16. Просим пользователя ввести верхнюю границу интегрирования и считываем ее в переменную b.
17. Вызываем функцию casemenu, передавая ей значения a, b, area и error.
18. Выводим на экран значение площади фигуры с помощью функции writeln и форматируем вывод как вещественное число с 2 знаками после запятой (area:0:2).
19. Выводим на экран оценку погрешности с помощью функции writeln и форматируем вывод как вещественное число с 2 знаками после запятой (error:0:2).
20. Происходит очистка окна с помощью команды window.Clear.
21. Затем вызывается функция draw\_plane, которая отрисовывает плоскость с заданным масштабом и размерами a и b.
22. Далее вызываются функции rects и graph, которые отрисовывают прямоугольники и график на плоскости с заданными параметрами.
23. Затем программа входит в цикл, где ожидает нажатия клавиш пользователем.
24. При нажатии клавиши, она считывается с помощью функции ReadKey и присваивается переменной ch.
25. Затем используется условный оператор case для выполнения определенных действий в зависимости от нажатой клавиши.
26. Если нажата клавиша стрелка вниз (#40), то происходит очистка окна, уменьшение масштаба(scale), и повторное отрисовывание плоскости, прямоугольников и графика.
27. Если нажата клавиша стрелка вверх (#38), то происходит очистка окна, увеличение масштаба(scale), и повторное отрисовывание плоскости, прямоугольников и графика.
28. Если пользователь выбрал пункт 2, выводим на экран "Программа завершена".
29. Если пользователь ввел некорректный выбор, выводим на экран "Неверный выбор. Попробуйте снова".
30. Переходим на новую строку.
31. Проверяем условие цикла until, если choice = 2, то выходим из цикла, иначе повторяем цикл с пунктом 3.

**Схема алгоритма с комментариями**

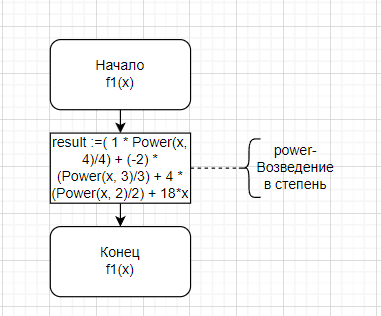


Рис.1 Схема алгоритма function f1(x).

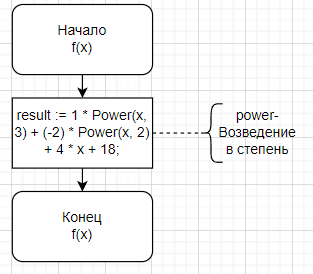
****

Рис.2 Схема алгоритма function f(x).

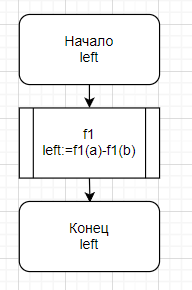


Рис.3 Схема алгоритма function left.

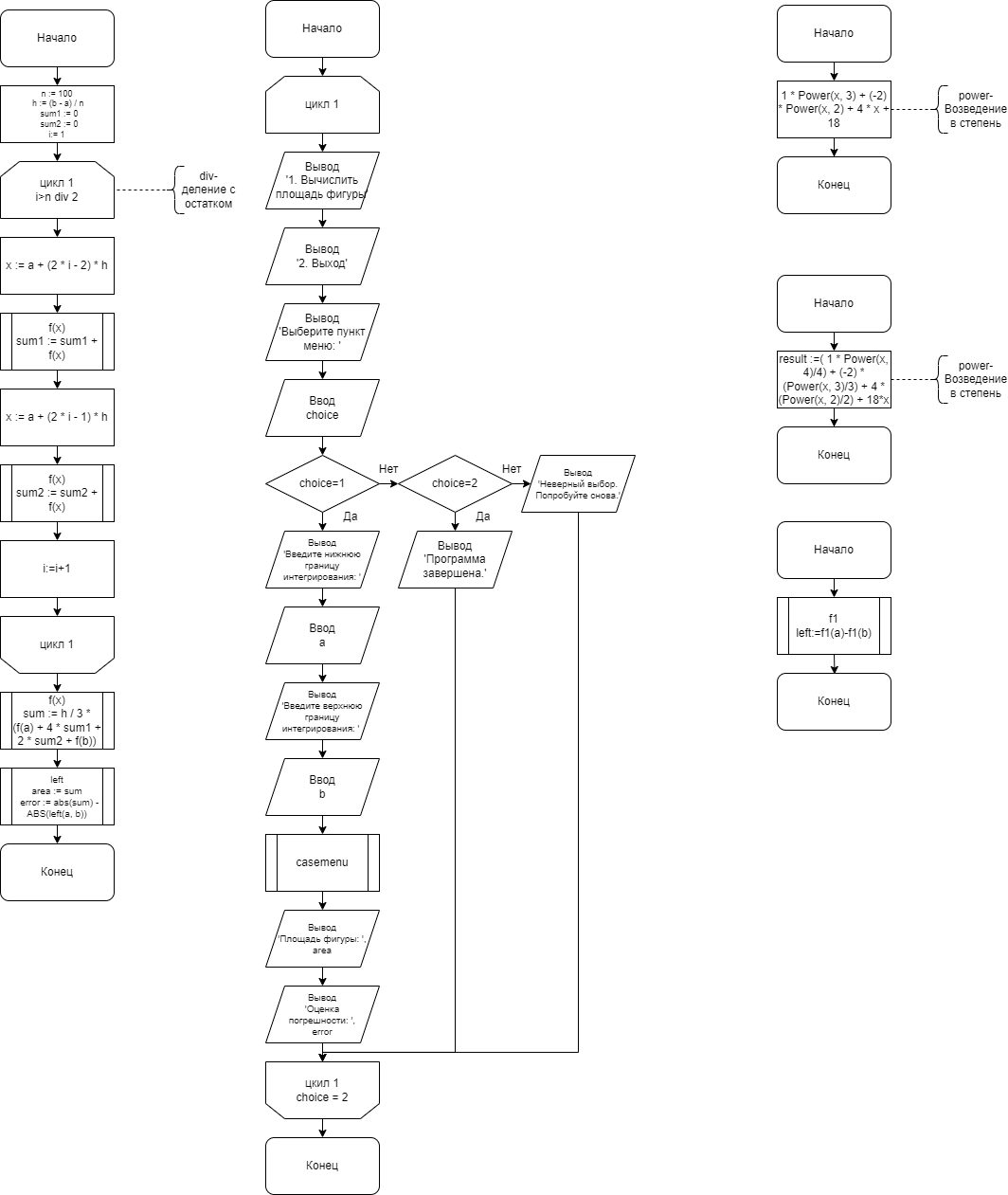


Рис.4 Схема алгоритма procedure casemenu.

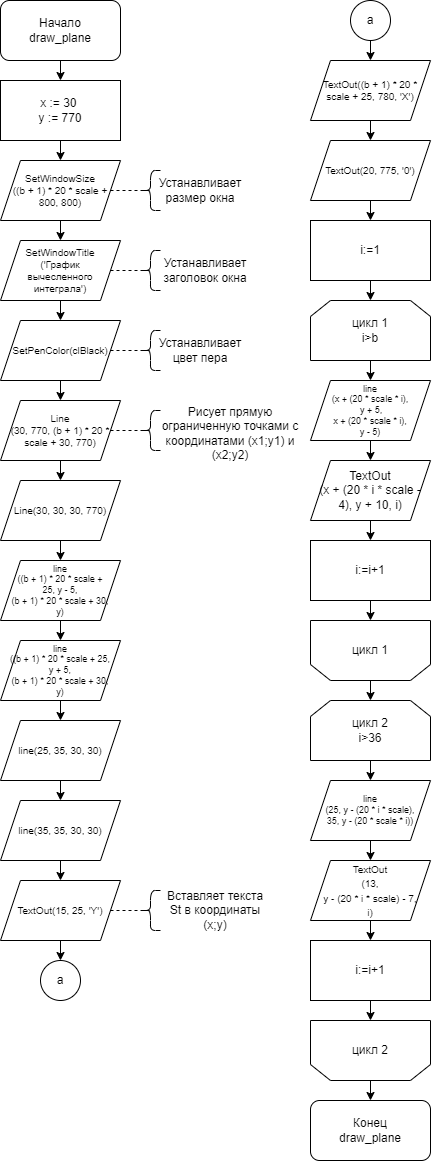


Рис.5 Схема алгоритма procedure draw\_plane.

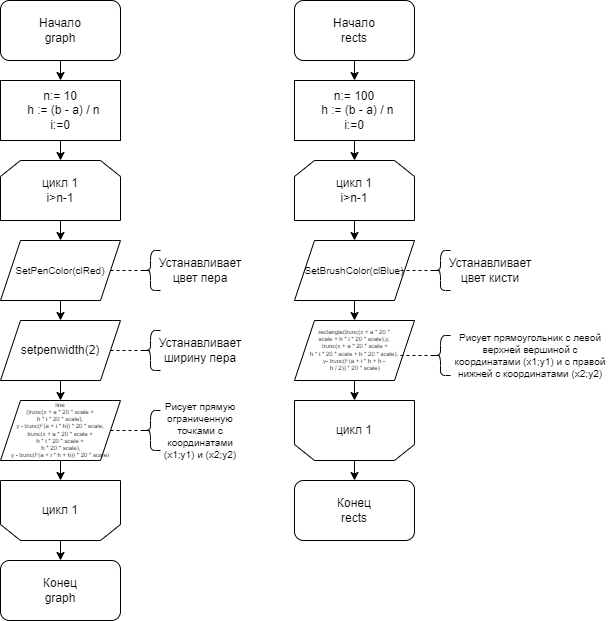


Рис.6 Схема алгоритма function graph.

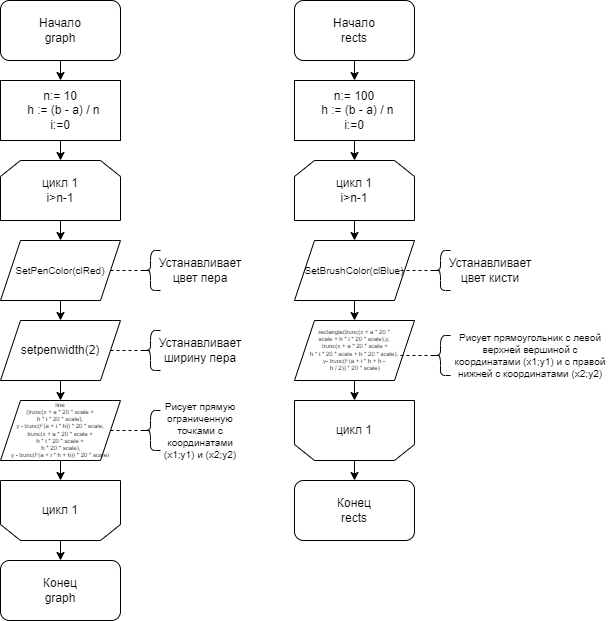


Рис.7 Схема алгоритма function rects.

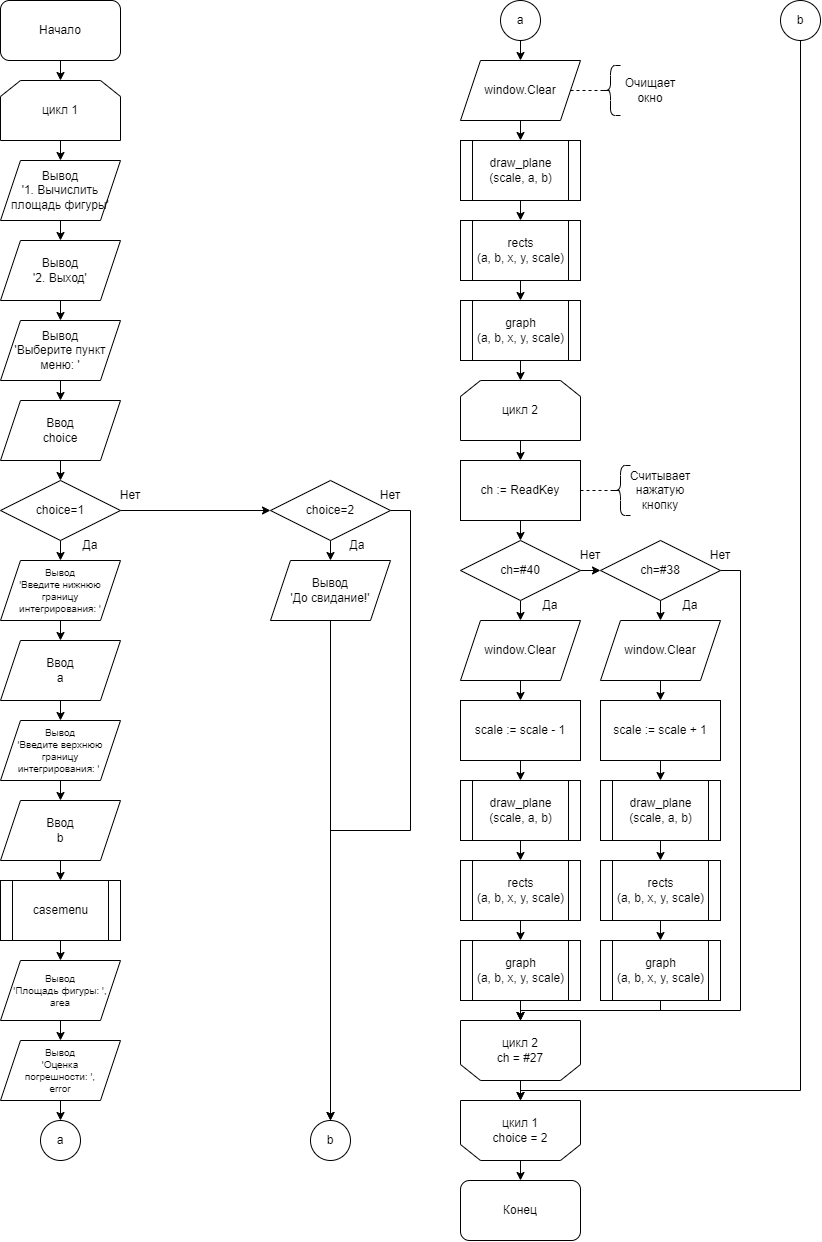


Рис.8 Схема алгоритма задачи.

**Код программы**

**uses** Crt, GraphABC;

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

**begin**

result := 1 \* Power(x, 3) + (-2) \* Power(x, 2) + 4 \* x + 18;

**end**;

**function** f1(x: real): real;

**begin**

result :=( 1 \* Power(x, 4)/4) + (-2) \* (Power(x, 3)/3) + 4 \* (Power(x, 2)/2) + 18\*x;

**end**;

**function** left(a, b:real):real;

**var**

h, x, sum:real;

**begin**

left:=f1(a)-f1(b);

**end**;

**procedure** casemenu (a, b: real; **var** area, error: real);

**var**

n, i: integer;

x, h: real;

sum1, sum2, sum: real;

**begin**

n := 100;

h := (b - a) / n;

sum1 := 0;

sum2 := 0;

**for** i := 1 **to** n **div** 2 **do**

**begin**

x := a + (2 \* i - 2) \* h;

sum1 := sum1 + f(x);

x := a + (2 \* i - 1) \* h;

sum2 := sum2 + f(x);

**end**;

sum := h / 3 \* (f(a) + 4 \* sum1 + 2 \* sum2 + f(b));

area := sum;

error := abs(sum) - ABS(left(a, b));

**end**;

**var**

S: Real;

a, b, n, x, y, option, scale: Integer;

ch: char;

area, error: Real;

**procedure** draw\_plane(scale, a, b: integer);

**var**

i, x, y: integer;

**begin**

x := 30;

y := 770;

SetWindowSize((b + 1) \* 20 \* scale + 800, 800);

SetWindowTitle('График вычесленного интеграла');

SetPenColor(clBlack);

Line(30, 770, (b + 1) \* 20 \* scale + 30, 770); // Ось X

Line(30, 30, 30, 770); // Ось Y

line((b + 1) \* 20 \* scale + 25, y - 5, (b + 1) \* 20 \* scale + 30, y);

line((b + 1) \* 20 \* scale + 25, y + 5, (b + 1) \* 20 \* scale + 30, y);

line(25, 35, 30, 30);

line(35, 35, 30, 30);

TextOut(15, 25, 'Y');

TextOut((b + 1) \* 20 \* scale + 25, 780, 'X');

TextOut(20, 775, '0');

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

**begin**

line(x + (20 \* scale \* i), y + 5, x + (20 \* scale \* i), y - 5);

TextOut(x + (20 \* i \* scale - 4), y + 10, i);

**end**;

**for** i := 1 **to** 36 **do**

**begin**

line(25, y - (20 \* i \* scale), 35, y - (20 \* scale \* i));

TextOut(13, y - (20 \* i \* scale) - 7, i);

**end**;

**end**;

**function** graph(a, b, x, y, scale: integer): boolean;

**var**

i: integer;

h: Real;

**begin**

n:= 10;

h := (b - a) / n;

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

**begin**

SetPenColor(clRed);

setpenwidth(2);

line(trunc(x + a \* 20 \* scale + h \* i \* 20 \* scale), y - trunc(F(a + i \* h)) \* 20 \* scale, trunc(x + a \* 20 \* scale + h \* i \* 20 \* scale + h \* 20 \* scale), y - trunc(F(a + i \* h + h)) \* 20 \* scale);

**end**;

**end**;

**function** rects(a, b, x, y, scale: integer): boolean;

**var**

i: integer;

h: Real;

**begin**

n:= 100;

h := (b - a) / n;

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

**begin**

SetBrushColor(clBlue);

rectangle(trunc(x + a \* 20 \* scale + h \* i \* 20 \* scale), y, trunc(x + a \* 20 \* scale + h \* i \* 20 \* scale + h \* 20 \* scale), y- trunc(F(a + i \* h + h - h / 2)) \* 20 \* scale);

**end**;

**end**;

**begin**

x := 30;

y := 770;

scale := 1;

SetConsoleIO;

**repeat**

Writeln('Выберите действие:');

Writeln('1. Рассчитать площадь фигуры');

Writeln('2. Выйти из программы');

Readln(option);

ClrScr;

**case** option **of**

1:

**begin**

n:=100;

Writeln('Введите пределы интегрирования a и b:');

Readln(a, b);

ClrScr;

casemenu(a, b, area, error);

Writeln('Площадь фигуры:', area:0:3);

Writeln('Вычисленная погрешность: ', error);

window.Clear;

draw\_plane(scale, a, b);

rects(a, b, x, y, scale);

graph(a, b, x, y, scale);

**repeat**

ch := ReadKey;

**begin**

ch := ReadKey;

**case** ch **of**

#40:

**begin**

window.Clear;

scale := scale - 1;

draw\_plane(scale, a, b);

rects(a, b, x, y, scale);

graph(a, b, x, y, scale);

**end**;

#38:

**begin**

window.Clear;

scale := scale + 1;

draw\_plane(scale, a, b);

rects(a, b, x, y, scale);

graph(a, b, x, y, scale);

**end**;

**end**;

**end**;

**until** ch = #27;

**end**;

2: Writeln('До свидание!');

**end**;

**until** option = 2;

**end**.

**Результат выполнения программы**

Пример 1. Выбираем пункт 1, вводим 1 и 3.

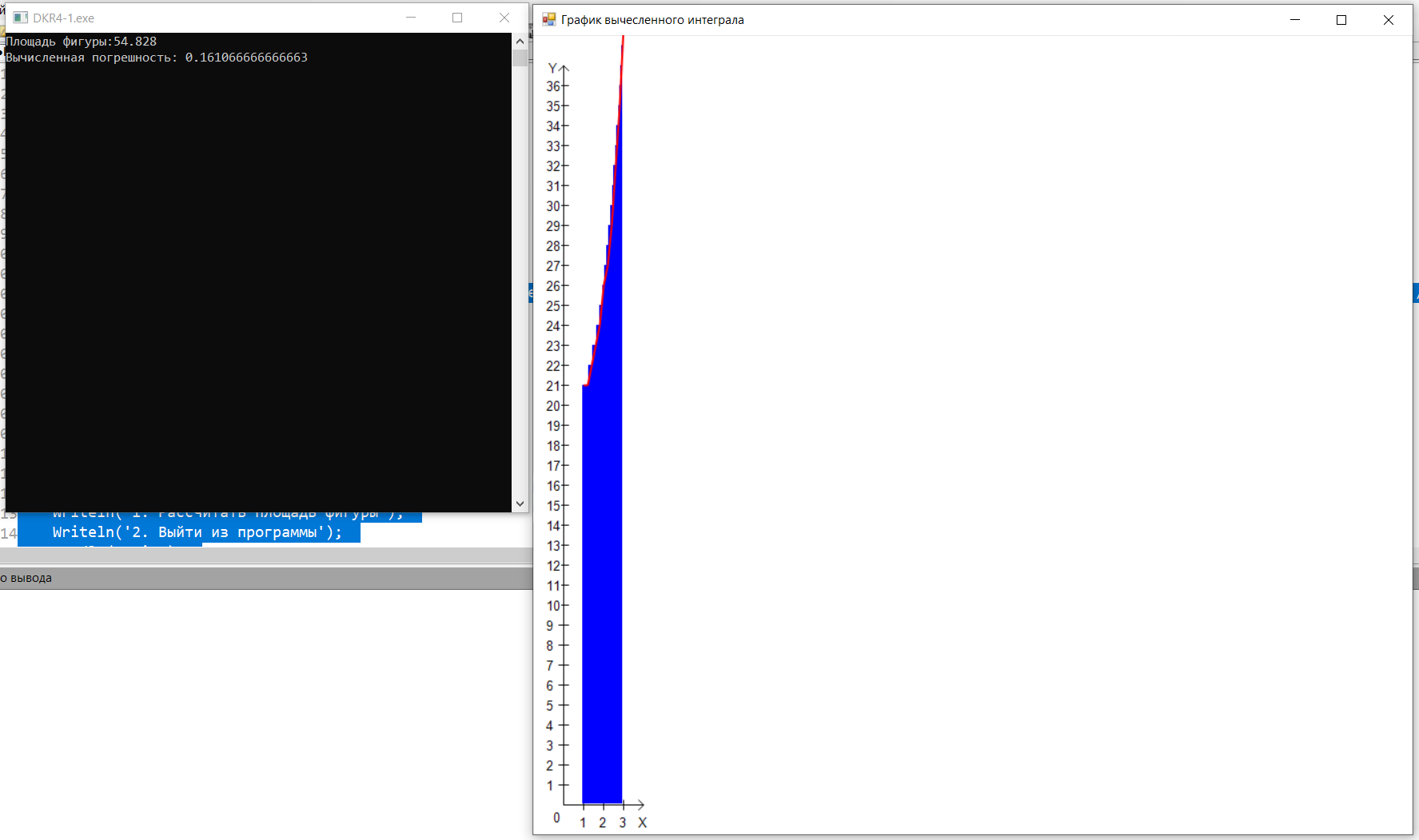


Рис.9 Результат выполнения программы (Пример 1).

Пример 2. Выбираем пункт 2.

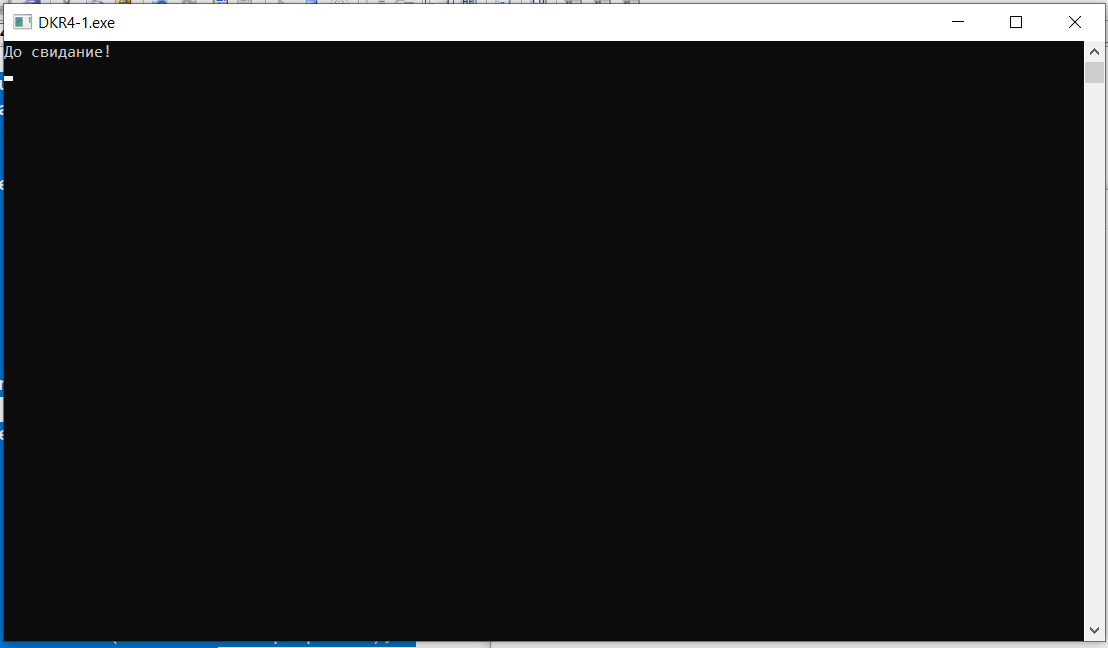


Рис.10 Результат выполнения программы (Пример 2).

Вывод: в результате выполнения домашней контрольной работы был освоен принцип работы в графическом режиме, получены базовые навыки взаимодействия с графическими примитивами.

Столкнулся с проблемой несовместимости модулей CRT и GraphABC. Для решения проблемы несовместимости модулей CRT и GraphABC, я изучил различные методы обхода данной проблемы. Один из таких методов включает изменение настроек компилятора, чтобы обеспечить совместимость модулей. Другим методом является использование альтернативных графических библиотек или модулей, которые совместимы с модулем CRT. В результате моих исследований и практического опыта, я научился успешно обходить проблему несовместимости модулей CRT и GraphABC. Теперь я могу без проблем работать в графическом режиме и использовать графические примитивы для создания интерактивных и креативных проектов. Преодоление подобных технических проблем во время домашних контрольных работ позволило мне развить свои навыки проблемного решения и настройки среды разработки. Теперь я более уверенно могу применять полученные навыки для решения других технических проблем, с которыми я могу столкнуться в будущем.