**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Вятский государственный университет»**

**Колледж ВятГУ**

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №3**

**«ИЗУЧЕНИЕ БАЗОВЫХ ПРИНЦИПОВ ОРГАНИЗАЦИИ ПРОЦЕДУР И ФУНКЦИЙ»**

**ПО ДИСЦИПЛИНЕ «ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»**

Выполнила: студент учебной группы

ИСПк-204-52-00

Бильчук Максим Алексеевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**Цель работы**

Освоить синтаксис построения процедур и функций, изучить способы передачи данных в подпрограммы, получить навыки организации минимального пользовательского интерфейса.

**Формулировка задания**

Вариант 24

1. Реализовать программу вычисления площади фигуры, ограниченной кривой 1 \* x ^ 3 + (1)\* x ^ 2 + (-4) \* x + (11) и осью OX (в положительной части по оси OY).

2. Вычисление определенного интеграла должно выполняться численно, с применением метода Симпсона

3. Пределы интегрирования вводятся пользователем.

4. Взаимодействие с пользователем должно осуществляться посредством сase-меню.

5. Требуется реализовать возможность оценки погрешности полученного результата.

6. Необходимо использовать процедуры и функции там, где это целесообразно.

**Описание алгоритма**

Для решения данной задачи используется модуль CRT для отображения консоли на экране пользователя, которая взаимодействует с самим пользователем. В меню case есть выбор из 2 пунктов, выбор которых осуществляется через case... выбор оператора. Чтобы перемещаться по меню, используйте процесс GoToXY для перемещения курсора в указанное положение на экране. Функция ReadKey используется для считывания клавиш с клавиатуры. Для визуального оформления контекстного меню используйте процесс textColor, чтобы изменить цвет текста.

Алгоритм использует метод Симпсона для вычисления определенного интеграла. Алгоритм использует функции и процедуры для вычисления производных, примитивов и фиксированных интегралов.

Когда будет выбрана опция «Начать интегрирование», пользователю будет предложено ввести предел интегрирования и количество интервалов, чтобы изменить точность вычисления конкретного интегрирования.

**Схема алгоритма с комментариями**

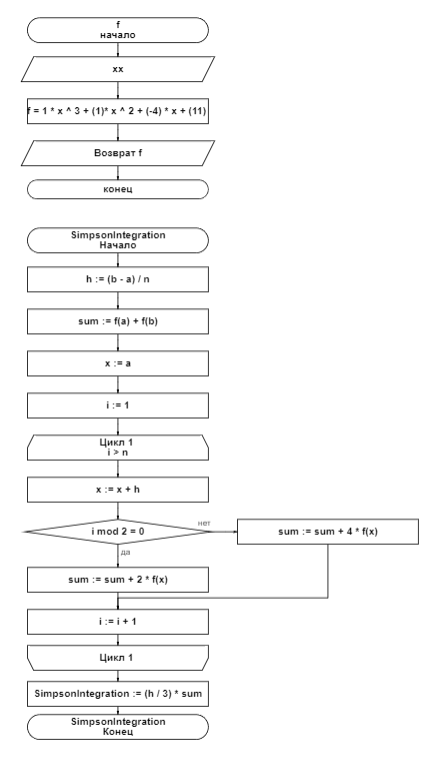


Рис. 1 «Схема алгоритма подпрограммы f and SimpsonIntegration»

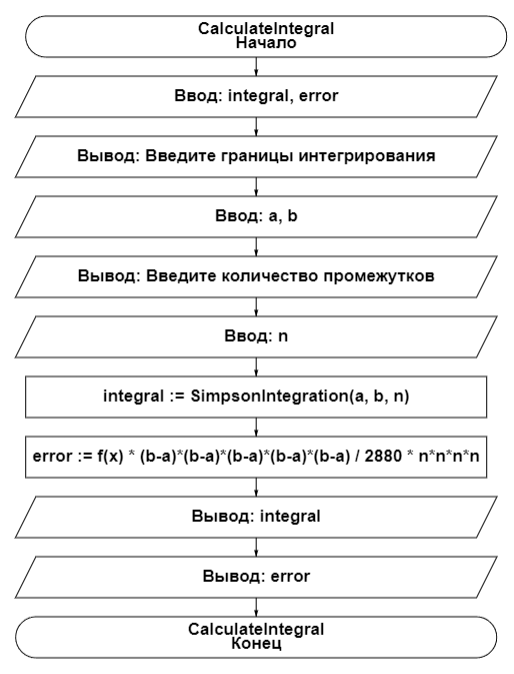


Рис. 2 «Схема алгоритма подпрограммы calculateintegral»

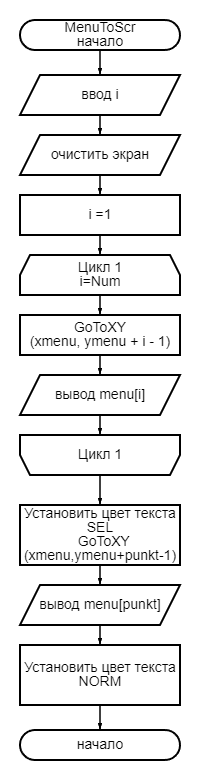
****

Рис. 3 «Схема алгоритма подпрограммы MenuToScr»

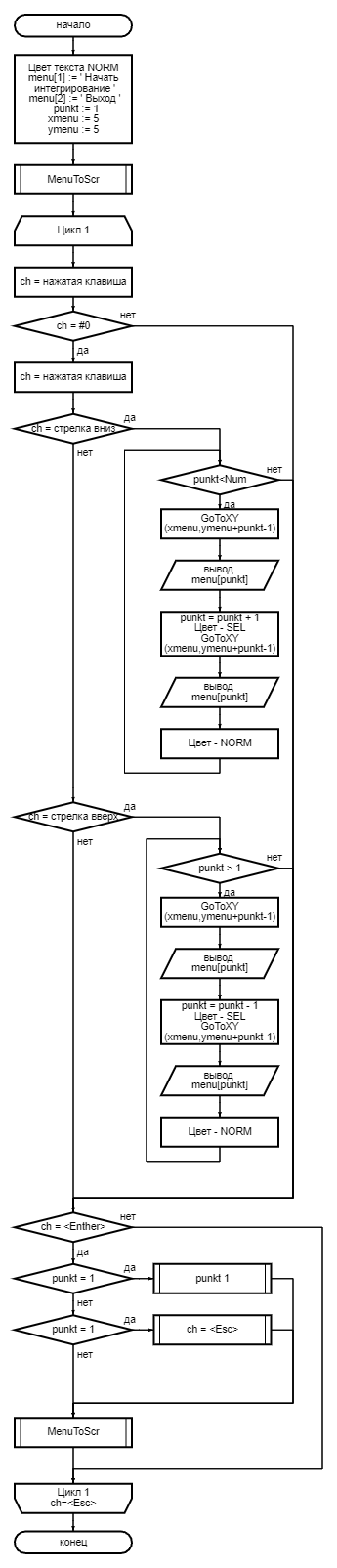
****

Рис. 4 «Схема алгоритма»

**Код программы**

**uses** Crt;

**const**

NORM = 12; // цвет невыделеного пункта

SEL = 10; // цвет выделенного пункта

Num = 2;

**var**

menu: **array**[1..Num] **of** string[24];// названия пунктов меню

punkt: integer;

ch: char;

xmenu, ymenu, TextAttr: byte;

a, b, c, x, d, ff: real;

n: integer;

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

**begin**

f := 1 \* xx \* xx \* xx + (1) \* xx \* xx + (-4) \* xx + 11;

**end**;

**function** SimpsonIntegration(a, b: Real; n: Integer): Real;

**var**

h, x, sum: Real;

i: Integer;

**begin**

h := (b - a) / n;

sum := f(a) + f(b);

x := a;

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

**begin**

x := x + h;

**if** i **mod** 2 = 0 **then**

sum := sum + 2 \* f(x)

**else**

sum := sum + 4 \* f(x);

**end**;

SimpsonIntegration := (h / 3) \* sum;

**end**;

**procedure** calculateIntegral;

**var**

integral, error: real;

**begin**

writeln('Введите границы интегрирования: ');

readln(a, b);

writeln('Введите количество промежутков: ');

readln(n);

integral := SimpsonIntegration(a, b, n);

error := f(x) \* (b-a)\*(b-a)\*(b-a)\*(b-a)\*(b-a) / 2880 \* n\*n\*n\*n;// Вычисление погрешности

writeln('Интеграл равен: ', integral:0:3);

writeln('Погрешность: ', error:0:3);

**end**;

**procedure** punkt1;

**begin**

ClrScr;

calculateIntegral;

writeln;

writeln('Процедура завершена вот такие пироги. Нажмите <Enter> для продолжения.');

**repeat**

ch := readkey;

**until** ch = #13;

**end**;

**procedure** MenuToScr;// вывод меню на экран

**var**

i: integer;

**begin**

ClrScr;

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

**begin**

GoToXY(xmenu, ymenu + i - 1);

write(menu[i]);

**end**;

TextColor(SEL);

GoToXY(xmenu, ymenu + punkt - 1);

write(menu[punkt]);// выделим строку в меню

TextColor(NORM);

**end**;

**begin**

menu[1] := ' Начать интегрирование ';

menu[2] := ' Выход ';

punkt := 1; xmenu := 5; ymenu := 5;

TextColor(NORM);

MenuToScr;

**repeat**

ch := ReadKey;

**if** ch = #0 **then begin**

ch := ReadKey;

**case** ch **of**

#40:// стрелка вниз

**if** punkt < Num **then begin**

GoToXY(xmenu, ymenu + punkt - 1); write(menu[punkt]);

punkt := punkt + 1;

TextColor(SEL);

GoToXY(xmenu, ymenu + punkt - 1); write(menu[punkt]);

TextColor(NORM);

**end**;

#38:// стрелка вверх

**if** punkt > 1 **then begin**

GoToXY(xmenu, ymenu + punkt - 1); write(menu[punkt]);

punkt := punkt - 1;

TextColor(SEL);

GoToXY(xmenu, ymenu + punkt - 1); write(menu[punkt]);

TextColor(NORM);

**end**;

**end**;

**end**

**else**

**if** ch = #13 **then begin**

**case** punkt **of**

1: punkt1;

2: ch := #27;// выход из программы

**end**;

MenuToScr;

**end**;

**until** ch = #27;

**end**.

**Результат выполнения программы**

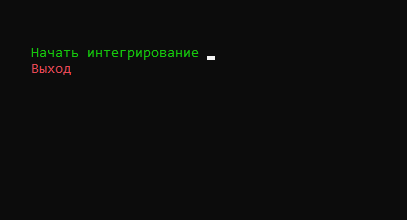


Рис. 5 «Case-меню»

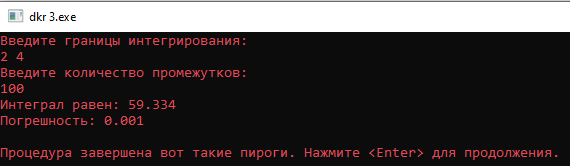


Рис. 6 «Результат интегрирования»

**Вывод**

В ходе данной домашней контрольной работы было изучено синтаксис процесса построения и функций, как передавать данные в подпрограммы, и приобретены навыки организации самого маленького пользовательского интерфейса.

Во время работы мы столкнулись с проблемами при написании кода или создании диаграмм алгоритмов. Некоторые из этих проблем заключаются в следующем::

У Pascal и Turbo Pascal разные коды клавиш, поэтому изначально использование клавиш со стрелками вверх и вниз для выбора пунктов меню не работало, но после поиска информации об этой проблеме это было исправлено.

Мы успешно справились с выполнением задания и были заинтересованы и увлечены процессом. Наше понимание использования процедур и функций было закреплено, и мы достигли всех поставленных целей.