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 ОДНОМЕРНЫХ МАССИВОВ И СТРОК»**

**ПО ДИСЦИПЛИНЕ «ИЗУЧЕНИЕ БАЗОВЫХ ПРИНЦИПОВ ОРГАНИЗАЦИИ ПРОЦЕНДУР И ФУНКЦИЙ»**

Выполнил: студент учебной группы

ИСПк-203-52-00

Шумилов Иван Андреевич

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**1.** **Цель работы:**

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

**2. Формулировка задания:**

**Вариант 24**

Задание состоит из шести частей:

1. Реализовать программу вычисления площади ограниченной кривой 2\*x^3+(0)\* x^2+(-1)\*x+(11) и осью OX (в положительной части по оси OY)

2. Вычисление определенного интеграла должно выполняться числено с применением метода Симпсона.

3. Пределы интегрирования вводятся пользователем

4. Взаимодействие с пользователем должно осуществляться посредством case-меню

5. Требуется реализовать возможность оценки погрешности полученного результата.

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

**3.1 Описание алгоритма:**

**1. Вычисление значения функции:**

Функция CalculateFunction выполняет вычисление значения функции. 2*x*3−*x*+11 для заданного значения x. Эта функция используется для определения поведения кривой, интеграл которой мы собираемся вычислить.

**2. Численное вычисление определенного интеграла:**

Процедура SimpsonMethod осуществляет численное вычисление определенного интеграла на заданном отрезке, используя метод Симпсона. Алгоритм разбивает отрезок на подотрезки, аппроксимирует функцию на каждом подотрезке, и затем объединяет результаты, учитывая веса для точек метода Симпсона.

**3. Ввод пределов интегрирования:**

Процедура InputLimits предоставляет пользователю возможность ввести нижний и верхний пределы интегрирования. Эти пределы определяют отрезок, на котором будет проведено численное интегрирование.

**4. Взаимодействие с пользователем через меню:**

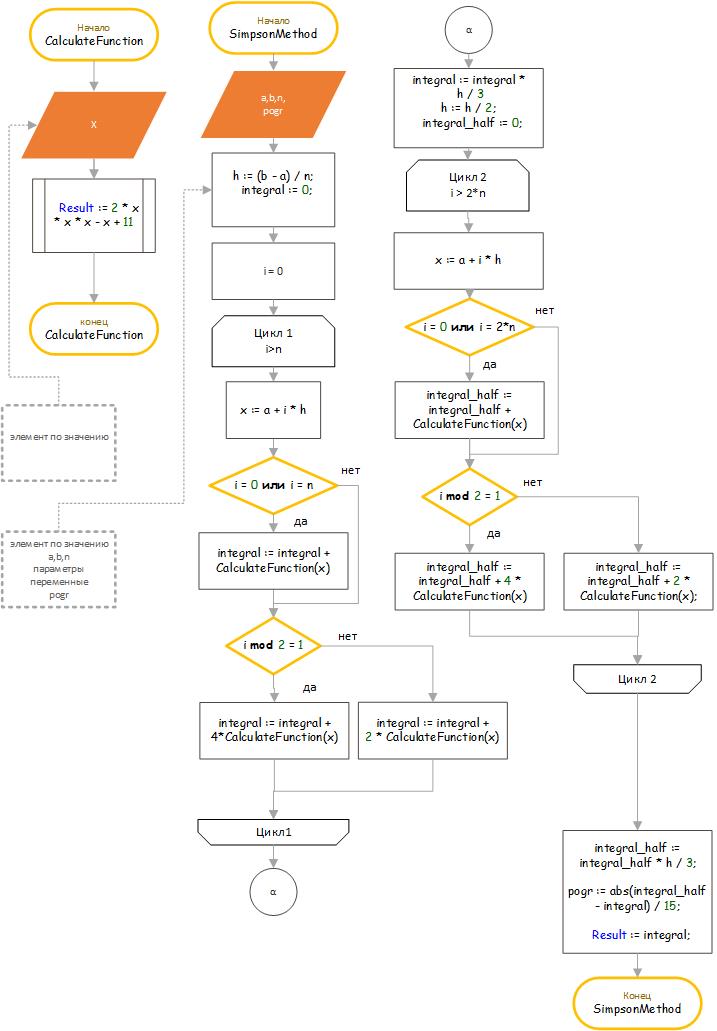
Программа использует case-меню для взаимодействия с пользователем и выбора действий. Пользователь может выбрать вычисление определенного интеграла или завершение программы.

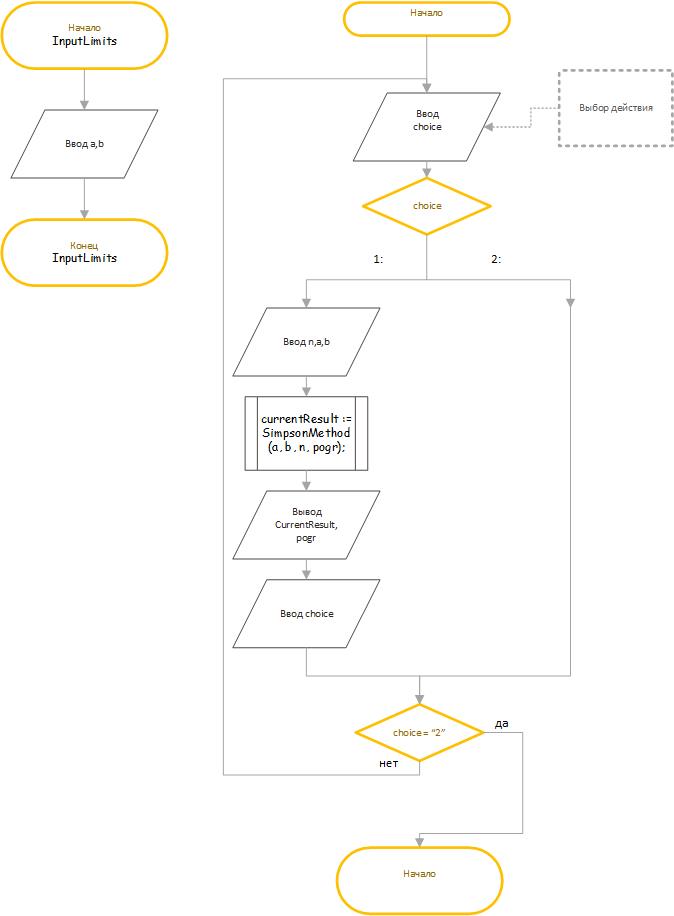
**5. Оценка погрешности:**

В коде предусмотрены комментарии и подсказки для внесения логики оценки погрешности. Предоставленный алгоритм оценки погрешности основан на сравнении результатов вычислений с разным размером шага интегрирования.

**6. Использование процедур и функций:**

Код структурирован с использованием процедур (SimpsonMethod, InputLimits) и функций (CalculateFunction), что способствует более четкому и организованному коду. Это соответствует целям использования процедур и функций, улучшая читаемость и переиспользуемость кода.

**4. Схема алгоритма с комментариями:** 



**5. Код программы:**

**program** Lab3;

**uses** crt;

**var**

a, b: real;

choice, n: integer;

// Функция для вычисления значения функции 2\*x^3 + 0\*x^2 - x + 11

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

**begin**

Result := 2 \* x \* x \* x - x + 11;

**end**;

// Функция для численного вычисления определенного интеграла методом Симпсона

**function** SimpsonMethod(a, b: real; n: integer; **var** pogr: real): real;

**var**

h, x, integral, integral\_half: real;

i: integer;

**begin**

// Вычисляем шаг интегрирования

h := (b - a) / n;

integral := 0;

// Выполняем метод Симпсона

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

**begin**

x := a + i \* h;

**if** (i = 0) **or** (i = n) **then**

integral := integral + CalculateFunction(x)

**else if** i **mod** 2 = 1 **then**

integral := integral + 4 \* CalculateFunction(x)

**else**

integral := integral + 2 \* CalculateFunction(x);

**end**;

// Умножаем результат на h/3, как требует метод Симпсона

integral := integral \* h / 3;

// Теперь вычисляем значение интеграла с шагом h/2

h := h / 2;

integral\_half := 0;

**for** i := 0 **to** 2 \* n **do**

**begin**

x := a + i \* h;

**if** (i = 0) **or** (i = 2 \* n) **then**

integral\_half := integral\_half + CalculateFunction(x)

**else if** i **mod** 2 = 1 **then**

integral\_half := integral\_half + 4 \* CalculateFunction(x)

**else**

integral\_half := integral\_half + 2 \* CalculateFunction(x);

**end**;

// Умножаем результат на h/3

integral\_half := integral\_half \* h / 3;

// Оцениваем погрешность

pogr := abs(integral\_half - integral) / 15;

Result := integral;

**end**;

// Процедура для ввода пределов интегрирования

**procedure** InputLimits(**var** a, b: real);

**begin**

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

readln(a);

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

readln(b);

**end**;

**begin**

**repeat**

// Выводим меню выбора

writeln('Выберите действие:');

writeln('1. Вычисление определенного интеграла');

writeln('2. Выход');

// Читаем выбор пользователя

readln(choice);

// Обрабатываем выбор

**case** choice **of**

1:

**begin**

// Выполняем вычисление интеграла

InputLimits(a, b);

writeln('Введите количество шагов для метода Симпсона: ');

readln(n);

// Вычисляем интеграл с текущим числом шагов и оцениваем погрешность

**var** currentResult, pogr: real;

currentResult := SimpsonMethod(a, b, n, pogr);

// Выводим результаты и погрешность

writeln('Результат вычисления интеграла: ', currentResult);

writeln('Погрешность: ', pogr);

**end**;

2:

writeln('Программа завершена.');

**end**;

**until** choice = 2; // Повторяем цикл, пока не выбран второй пункт (выход)

**end**.

**6.1. Результат выполнения программы**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

**7.**  **Вывод:**

В ходе выполнения работы мы изучили принципы создания процедур и функций на языке Pascal, а также освоили способы передачи данных в подпрограммы. Наша цель заключалась в овладении синтаксисом построения этих элементов программы и в получении опыта работы с базовым пользовательским интерфейсом.

Для решения задачи численного интегрирования мы выбрали метод Симпсона. Результатом стало создание программы, позволяющей пользователю ввести параметры интегрирования: нижний и верхний пределы, а также количество шагов метода Симпсона.

Одним из ключевых моментов работы стало внимание, уделенное оценке погрешности. Мы применили простой метод сравнения результатов на разных шагах, что позволило нам получить представление о точности численного метода.

Таким образом, мы успешно справились с задачей, закрепили навыки построения процедур и функций, а также научились создавать минимальный, но функциональный пользовательский интерфейс. Эти навыки могут быть полезны в будущем при решении более сложных задач на языке Pascal.