ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Метод прямоугольников

Студент**:** Назирджанов Н.Ф

Преподаватель: Перл О.В.

Санкт-Петербург
2023г.

Описание метода

1. Необходимо задать функцию, которую нужно проинтегрировать, а также интервал интегрирования, то есть начальную и конечную точки отрезка.

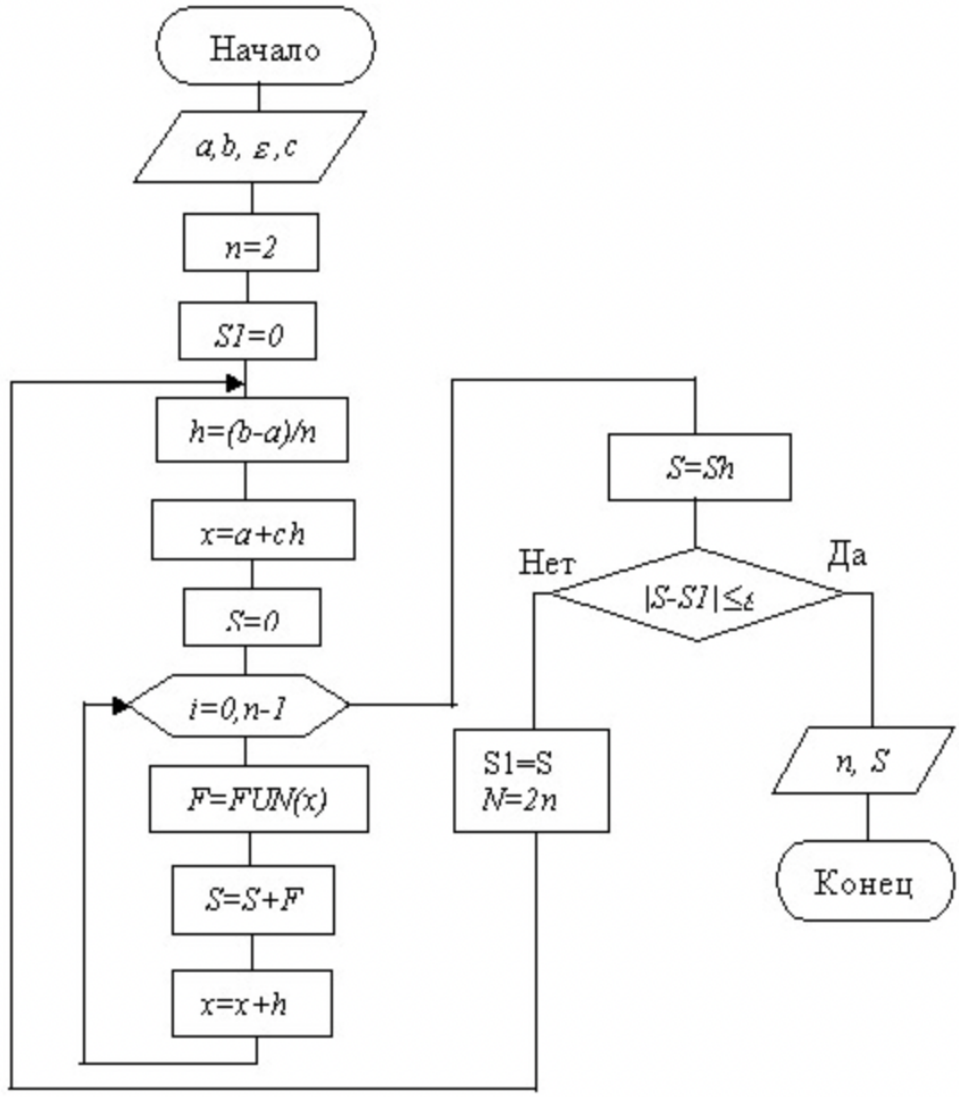
2. Разбить отрезок интегрирования на n равных частей (или прямоугольников). Для этого можно вычислить ширину каждого прямоугольника, разделив длину отрезка на n.

3. Для каждого прямоугольника выбрать точку на нем для вычисления значения функции. В методе средних прямоугольников это будет середина каждого прямоугольника, а в методе левых и правых прямоугольников - соответственно, левая или правая граница каждого прямоугольника.

4.Вычислить площадь каждого прямоугольника, умножив его ширину на значение функции в выбранной точке.

5. Сложить площади всех прямоугольников, чтобы получить приближенное значение интеграла.

Блок-схема



a,b - концы интервала,

ε- заданная точность,

c = 0 - метод левых прямоугольников,

c = 1 - метод правых прямоугольников,

S1 - значение интеграла на предыдущем шаге,

S - значение интеграла на текущем шаге.

Реализация:

```java
public class RectangleMethod {
    Functions functions = new Functions();
    OutputFunctions outputFunctions = new OutputFunctions();

    public void startMethodMid(double a, double b, double e, int number) {
        if (a > b) {
            double tmp = a;
            a = b;
            b = tmp;
        }
        double aNew = a, step, sum = 0, r = e + 1, I = functions.getI(a, b, number);
        long n = 2;

        while (r > e) {
            n *= 2;
            step = (b - aNew) / n;
            sum = 0;
            for (int i = 0; i < n; i++) {
                sum += functions.f(aNew + step / 2, number);
                aNew += step;
            }
            sum = sum * step;
            r = Math.abs(I - sum);
            aNew = a;
        }

        System.out.println("\nРешение методом средних прямоугольников:");

        if (Double.isNaN(sum) || Double.isNaN(I) || Double.isNaN(r) || Double.isNaN(Math.abs(100 * r / ((I + sum) / 2)))) {
            System.out.println("В выбранном интервале присутсвует разрыв первого рода!\n");
        } else {
            outputFunctions.outAnswer(e, sum, I, r, n);
        }
    }

    public void startMethodLeft(double a, double b, double e, int number) {
        if (a > b) {
            double tmp = a;
            a = b;
            b = tmp;
        }
        double aNew = a, step, sum = 0, r = e + 1, I = functions.getI(a, b, number);
        long n = 2;
        while (r > e) {
            n *= 2;
            step = (b - aNew) / n;
            sum = 0;
            for (int i = 0; i < n; i++) {
                sum += functions.f(aNew, number);
                aNew += step;
            }
            sum = sum * step;
            r = Math.abs(I - sum);
            aNew = a;
        }

        System.out.println("\nРешение методом левых прямоугольников:");

        if (Double.isNaN(sum) || Double.isNaN(I) || Double.isNaN(r) || Double.isNaN(Math.abs(100 * r / ((I + sum) / 2)))) {
            System.out.println("В выбранном интервале присутсвует разрыв первого рода!\n");
        } else {
            outputFunctions.outAnswer(e, sum, I, r, n);
        }
    }

    public void startMethodRight(double a, double b, double e, int number) {
        if (a > b) {
            double tmp = a;
            a = b;
            b = tmp;
        }
        double aNew = a, step, sum = 0, r = e + 1, I = functions.getI(a, b, number);
        long n = 2;

        while (r > e) {
            n *= 2;
            step = (b - aNew) / n;
            aNew += step;
            sum = 0;
            for (int i = 0; i < n; i++) {
                sum += functions.f(aNew, number);
                aNew += step;
            }
            sum = sum * step;
            r = Math.abs(I - sum);
            aNew = a;
        }

        System.out.println("\nРешение методом правых прямоугольников:");

        if (Double.isNaN(sum) || Double.isNaN(I) || Double.isNaN(r) || Double.isNaN(Math.abs(100 * r / ((I + sum) / 2)))) {
            System.out.println("В выбранном интервале присутсвует разрыв первого рода!\n");
        } else {
            outputFunctions.outAnswer(e, sum, I, r, n);
        }
    }
}

                                        //In programming, there is no place for magic, only intuition and logic
```

Для sin(x) + cos(x)

```
Выбирите метод:
1. Правых прямоугольников
2. Левых прямоугольников
3. Средних прямоугольников
4. Все методы
номер метода: 3
Введите нижний целочисленный предел интегрирования a: -1
Введите верхний целочисленный предел интегрирования b: 1
Введите точность e: 0,01

Решение методом средних прямоугольников:

----------------------------------------
Достигнута заданная точность e = 0.01
Calculated I = 1,68733
Standard I = 1,68294
Абсолютная погрешность |R| = 0,00439
Относительная погрешность E = 0,26055%
n = 8
```

Вывод:

Часто на практике не удается вычислить интеграл аналитическим путем. В этих случаях применяют приближенные методы численного интегрирования.

Метод прямоугольника, метод трапеций и метод Симпсона - это методы численного интегрирования, которые используются для приближенного вычисления определенных интегралов.

Метод прямоугольников : этот метод основан на приближенном представлении подынтегральной функции на каждом интервале интегрирования в виде прямоугольника, площадь которого вычисляется как произведение высоты прямоугольника (значения функции в любой точке на данном интервале) на ширину интервала. Затем суммируются площади всех прямоугольников, что дает приближенное значение интеграла.

Метод прямоугольников часто используется, когда нужно быстро оценить интеграл функции на отрезке с небольшой точностью. Он особенно полезен для функций с небольшим числом экстремумов и медленно изменяющихся функций.Также метод прямоугольников может использоваться в качестве первого приближения для более точных методов интегрирования, таких как метод трапеций или метод Симпсона.Однако, если функция имеет быстрое изменение на отрезке интегрирования или большое число экстремумов, метод прямоугольников может давать неприемлемо неточный результат. В таких случаях стоит использовать более точные методы, например, метод трапеций или метод Симпсона.