КОМИТЕТ ПО ОБРАЗОВАНИЮ ПРАВИТЕЛЬСТВА САНКТ-ПЕТЕРБУРГА

Санкт-Петербургское государственное

бюджетное профессиональное образовательное учреждение

«Колледж информационных технологий»

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3**

**по численным методам**

**Специальность 09.02.07**

**«Информационные системы и программирование»**

**Специализация**

**«Разработчик web и мультимедийных приложений»**

**(по программе базовой подготовки)**

Выполнил студент 91 гр.: А.В. Виноградов\_\_\_\_\_\_\_\_\_

Тел.

e-mail

Принял преподаватель: И.П. Смирнова \_\_\_\_\_\_\_\_\_\_

Санкт-Петербург 2020

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc20467494)

**ПРАКТИЧЕСКАЯ РАБОТА** 4

**ВВЕДЕНИЕ**

Задача – вычислить определенный интеграл тремя различными способами:

1. Вычислить интеграл методом средних прямоугольников (текст программы на языке программирования +.exe файл или скриншот с ответом);
2. Вычислить интеграл методом трапеций (текст программы на языке программирования + .exe файл или скриншот с ответом);
3. Вычислить интеграл методом Монте-Карло двумя способами:
   1. Вычислить интеграл методом Монте-Карло **программно** (текст программы на языке программирования + .exe файл или скриншот с ответом);
   2. Вычислить интеграл методом Монте-Карло **графически** (в электронных таблицах);
4. Сравнить полученные результаты.

Для проверки графического метода использовалась программа MS Excel. Для программной проверки использовался язык C, написание кода происходило в Visual Studio Code, проверка работы программы происходила в командной строке Visual Studio Code.

**ПРАКТИЧЕСКАЯ РАБОТА**

Сначала пробуем метод средних прямоугольников. Скриншот программы (рисунок 1):

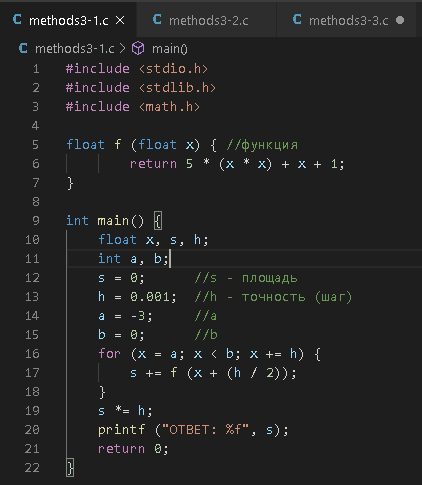


Рисунок 1 – Код программы

Текст программы:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

float f (float x) { //функция

        return 5 \* (x \* x) + x + 1;

}

int main() {

    float x, s, h;

    int a, b;

    s = 0;      //s - площадь

    h = 0.001;  //h - точность (шаг)

    a = -3;     //a

    b = 0;      //b

    for (x = a; x < b; x += h) {

        s += f (x + (h / 2));

    }

    s \*= h;

    printf ("OTBET: %f", s);

    return 0;

}

Результат работы программы в командной строке Visual Studio Code (рисунок 2):

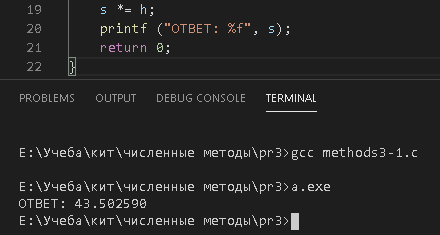


Рисунок 2 – Результат работы

Запоминаем ответ. Теперь попробуем метод трапеций (рисунок 3):

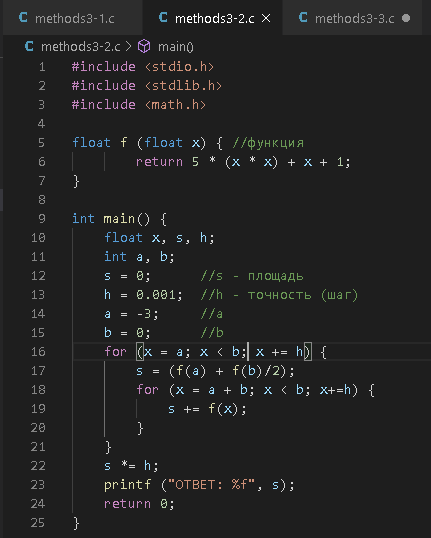


Рисунок 3 – Код программы

Текст программы:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

float f (float x) { //функция

        return 5 \* (x \* x) + x + 1;

}

int main() {

    float x, s, h;

    int a, b;

    s = 0;      //s - площадь

    h = 0.001;  //h - точность (шаг)

    a = -3;     //a

    b = 0;      //b

    for (x = a; x < b; x += h) {

        s = (f(a) + f(b)/2);

        for (x = a + b; x < b; x+=h) {

            s += f(x);

        }

    }

    s \*= h;

    printf ("OTBET: %f", s);

    return 0;

}

Результат работы программы в командной строке Visual Studio Code (рисунок 4):

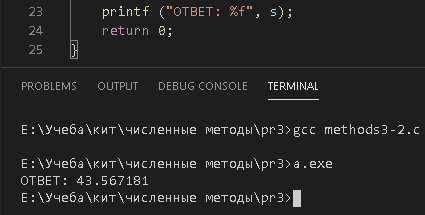


Рисунок 4 – Результат работы

Видно, что ответ стал более точным. Теперь посмотрим метод Монте-Карло (рисунок 5):

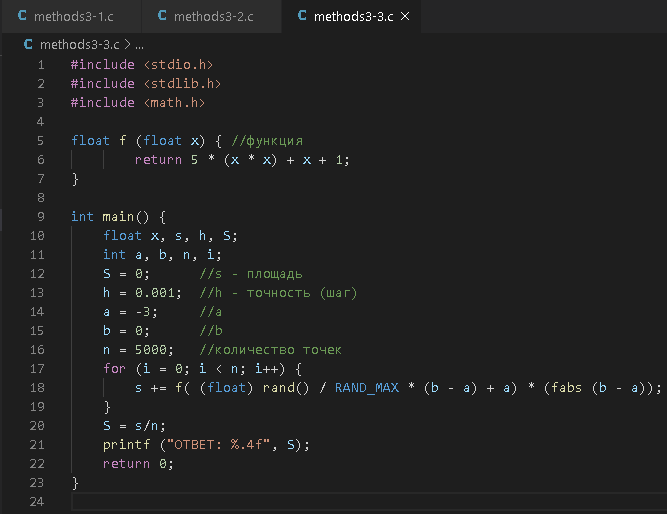


Рисунок 5 – Код программы

Текст программы:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

float f (float x) { //функция

        return 5 \* (x \* x) + x + 1;

}

int main() {

    float x, s, h, S;

    int a, b, n, i;

    S = 0;      //s - площадь

    h = 0.001;  //h - точность (шаг)

    a = -3;     //a

    b = 0;      //b

    n = 5000;   //количество точек

    for (i = 0; i < n; i++) {

        s += f( (float) rand() / RAND\_MAX \* (b - a) + a) \* (fabs (b - a));

    }

    S = s/n;

    printf ("OTBET: %.4f", S);

    return 0;

}

Результат работы программы в командной строке Visual Studio Code (рисунок 6):

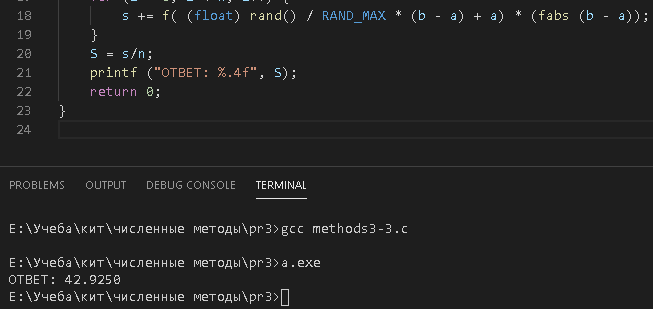


Рисунок 6 – Результат работы

Мы должны понимать, что метод Монте-Карло больше подходит для поиска площадей сложных фигур (к примеру, 100 окружностей разных размеров, случайно вписанных друг в друга), но при наличии более систематических методов он будет проигрывать в точности. При увеличении числа точек в 20 раз результат будет точнее (рисунок 7), но мы затрачиваем намного больше вычислительных мощностей, чем в первых двух методах, которые показывают более точные результаты.

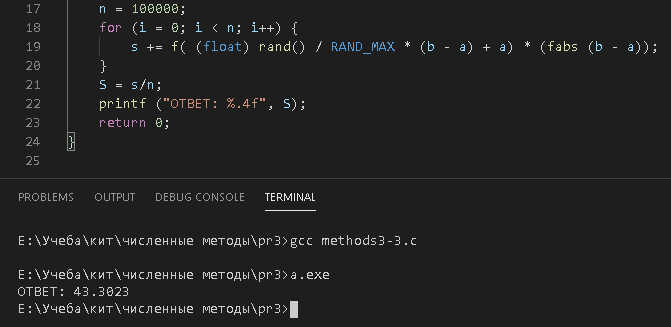


Рисунок 7 – Результат работы с большим кол-вом точек

Теперь посмотрим этот же метод, но выполненный в MS Excel (рисунок 8).

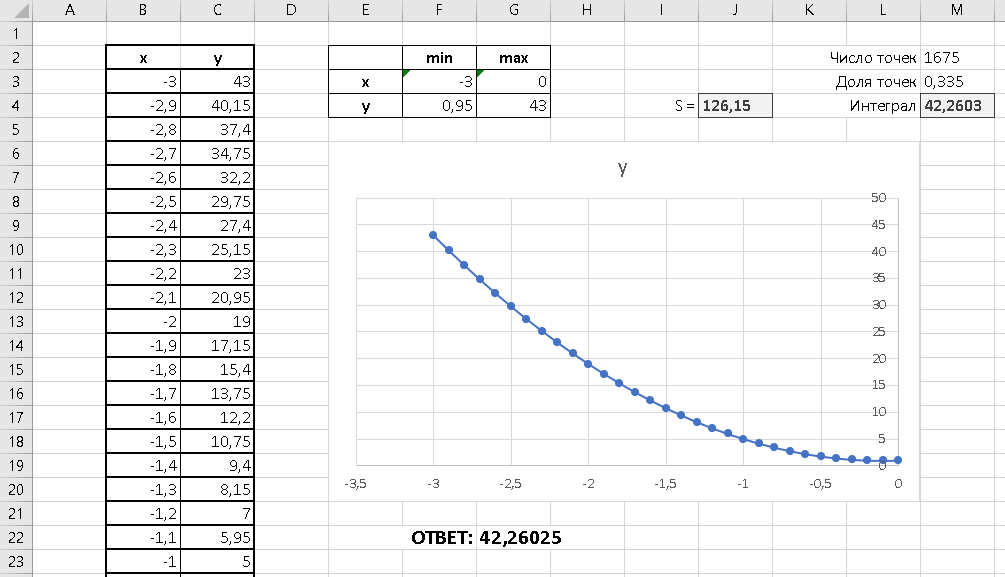


Рисунок 8 – Результат работы в Excel

Сравним результаты:

|  |  |
| --- | --- |
| 43.502590 | метод средних прямоугольников |
| 43.567181 | метод трапеций |
| 42.9250 | метод Монте-Карло программный (n = 5.000) |
| 43.3023 | метод Монте-Карло программный (n = 100.000) |
| 42.26025 | метод Монте-Карло графический (n = 5.000) |

Вывод – метод Монте-Карло является аналитическим инструментом, точный результат можно будет получить только с затрачиванием больших вычислительных мощностей. Для получения точных результатов им следует пользоваться **только** в том случае, если использование других способов не представляется возможным.