**Лабораторная работа 1**

**Численное интегрирование**

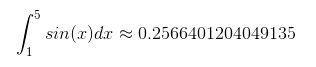
**при помощи информационных технологий**

**Язык:** С++14

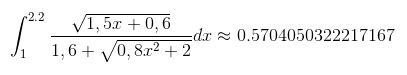
**Задача**: Написать программу, в которой реализуются методы численного интегрирования с постоянным и переменным шагом, а также метод двойного интегрирования.

**Математическая модель:**

Табличный интеграл:



Произвольный интеграл:



Двойной интеграл:



**Переменные:**

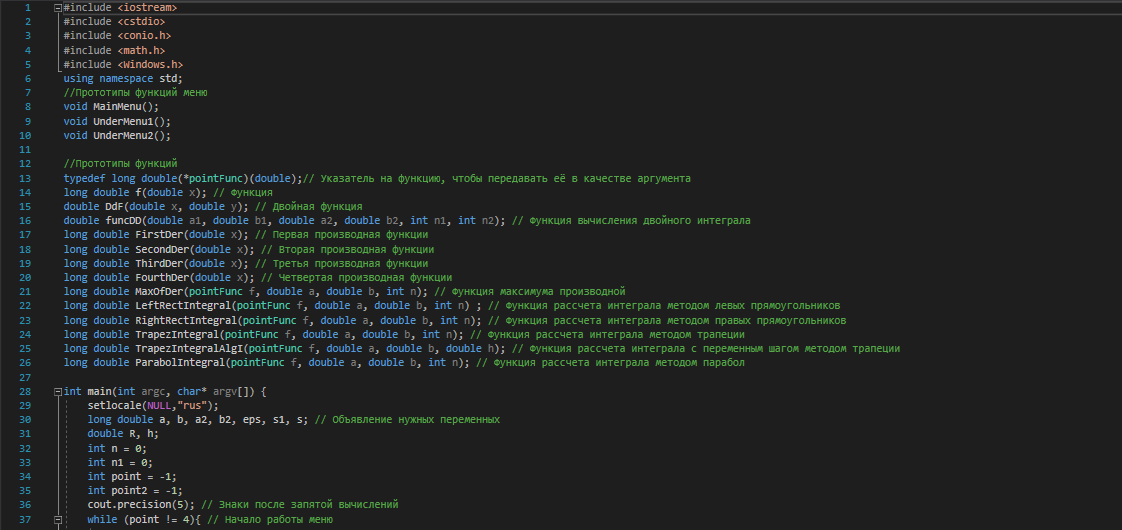
|  |  |  |
| --- | --- | --- |
| **Имя** | **Смысл** | **Тип** |
| MainMenu | Прототип, выводящий на экран главное меню | void |
| UnderMenu1 | Прототип, выводящий на экран первое подменю. | void |
| UnderMenu2 | Прототип, выводящий на экран второе подменю. | void |
| f | Функция вычисляет и возвращает значение заданного интеграла в точке x | long double |
| DdF | Функция вычисляет и возвращает значение двойного интеграла в точках x и y | double |
| funcDD | Функция, которая вычисляет двойной интеграл, после чего возвращает полученное значение. | double |

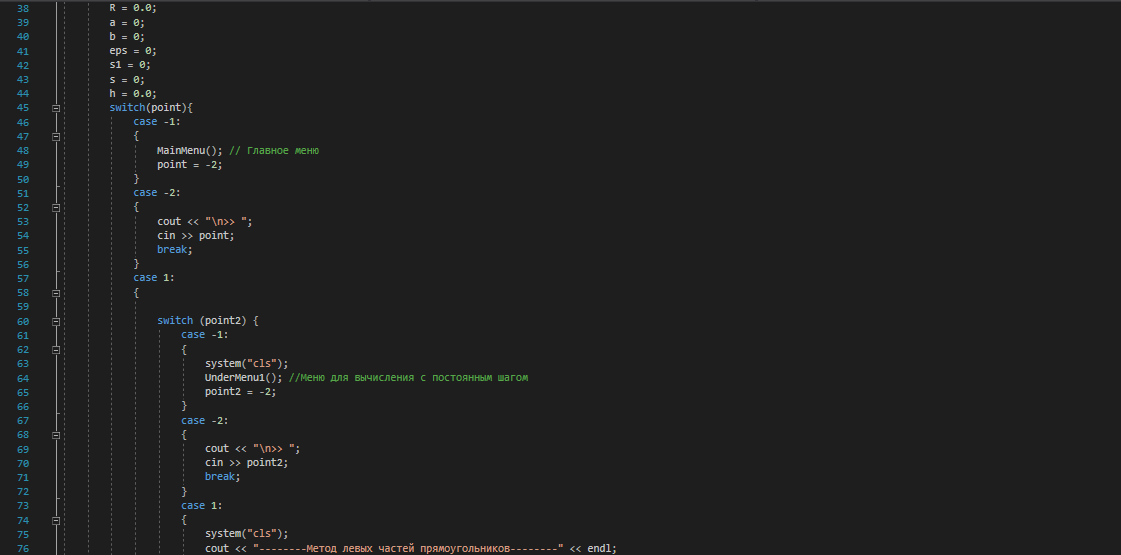
|  |  |  |
| --- | --- | --- |
| FirstDer | Функция вычисляет и возвращает значение заданной производной | long double |
| SecondDer |
| FourthDer |
| MaxOfDer | Функция находит максимальное значение производной и возвращает его. | long double |
| LeftRectIntegral | Функция расчета интегралов методом левых прямоугольников | long double |
| RightRectIntegral | Функция расчета интегралов методом правых прямоугольников | long double |
| TrapezIntegral | Функция расчета интегралов методом трапеций | long double |
| TrapezIntegralAlgI | Функция расчета интегралов с переменным шагом методом трапеций | long double |
| ParabolIntegral | Функция расчета интегралов методом парабол | long double |

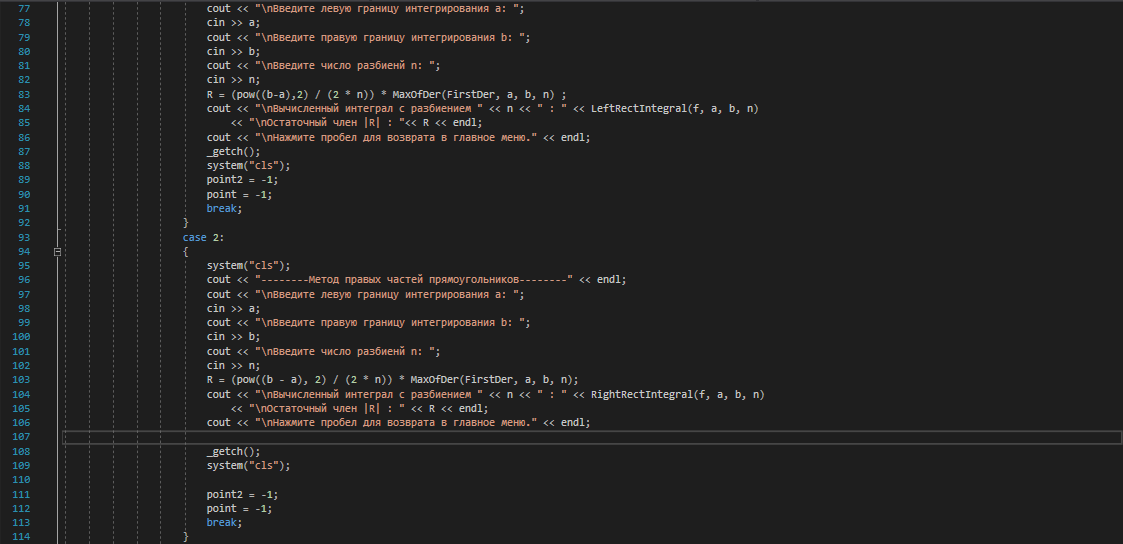
Основные переменные:

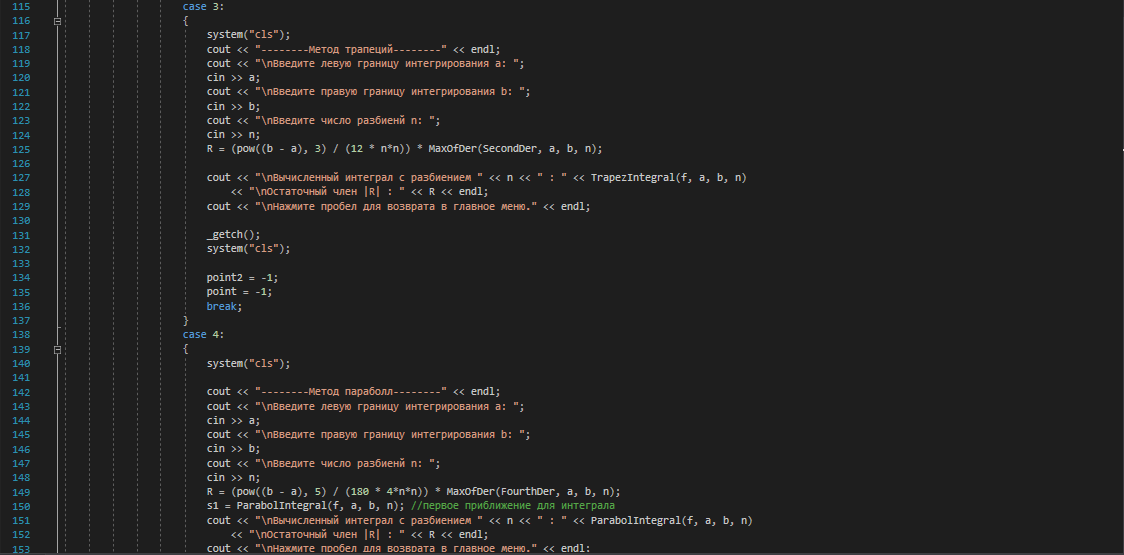
|  |  |  |
| --- | --- | --- |
| a, b, a1, b1 | Переменные, принимающие границы интегрирования | long double |
| eps | Переменная точности | long double |
| s, s1 | Переменные накопления сумм | long double |
| R | Переменная содержащая остаточный член | double |
| h | Переменная содержит шаг интегрирования | double |
| n, n1 | Переменные, принимающие число разбиений интегрируемых отрезков | int |
| point, point2 | Переменные для работы меню | int |

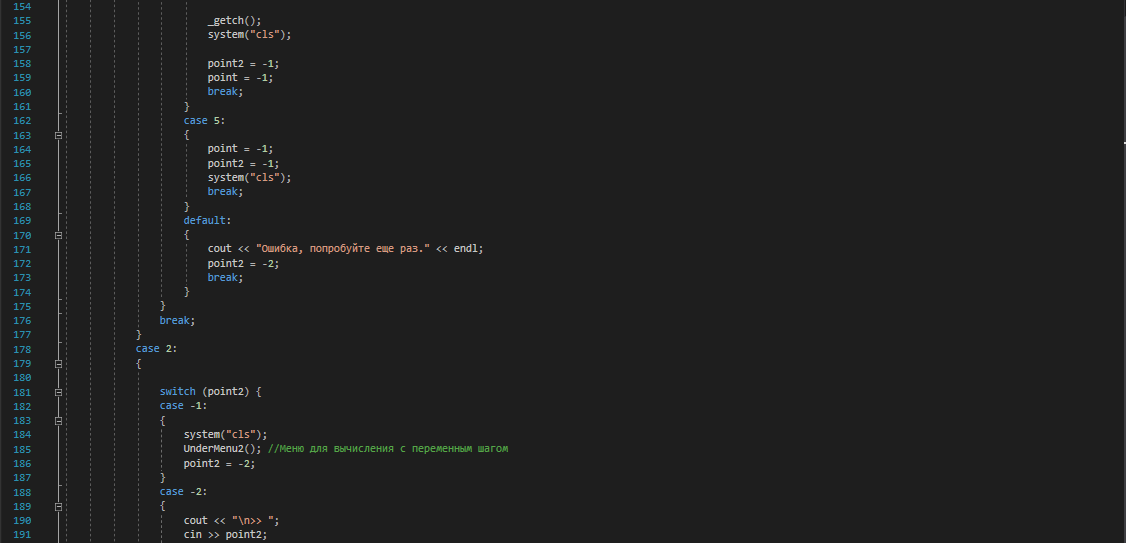
**Код программы:**

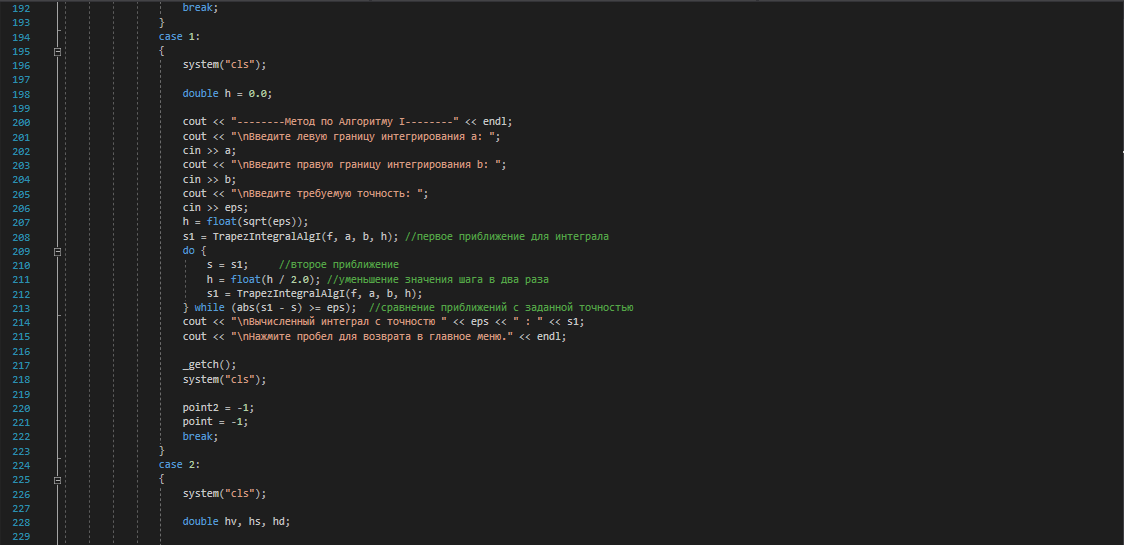


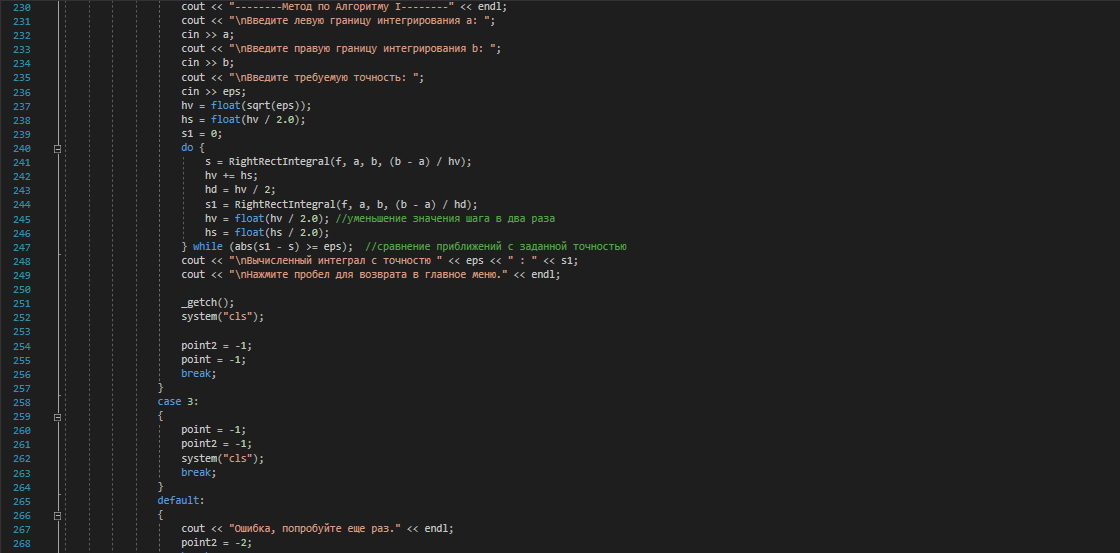




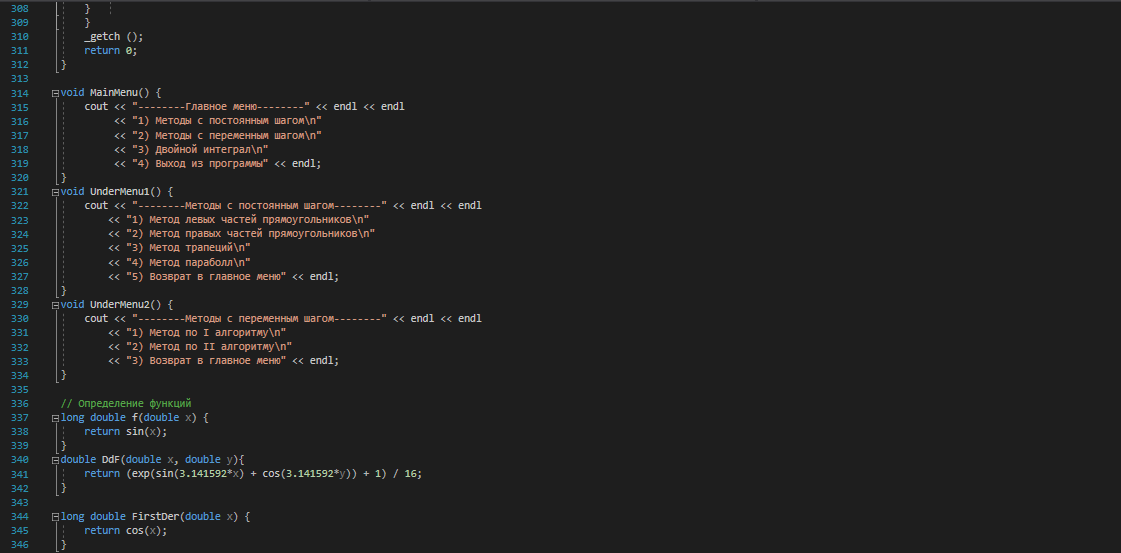




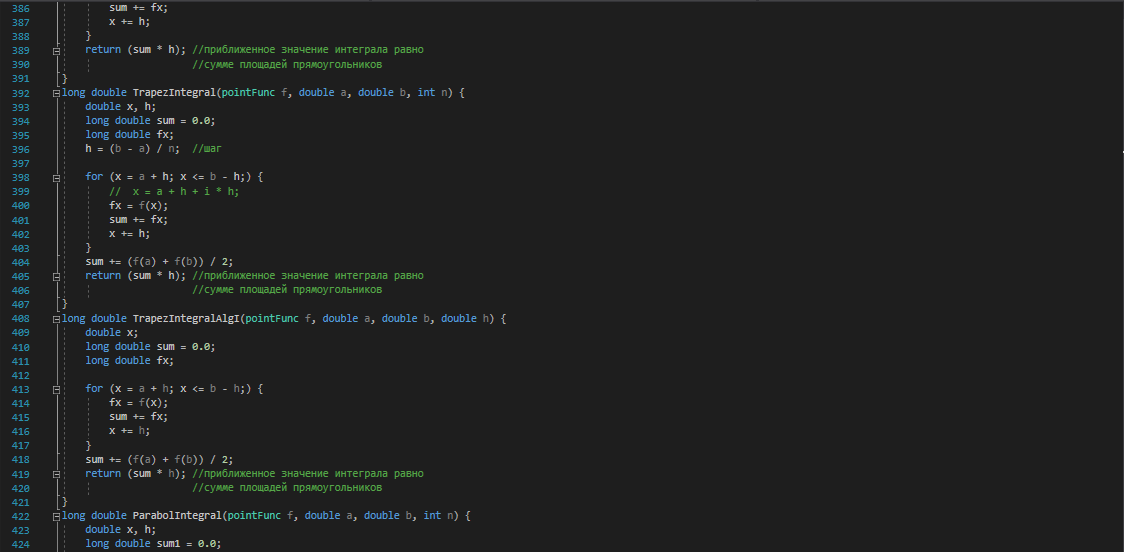


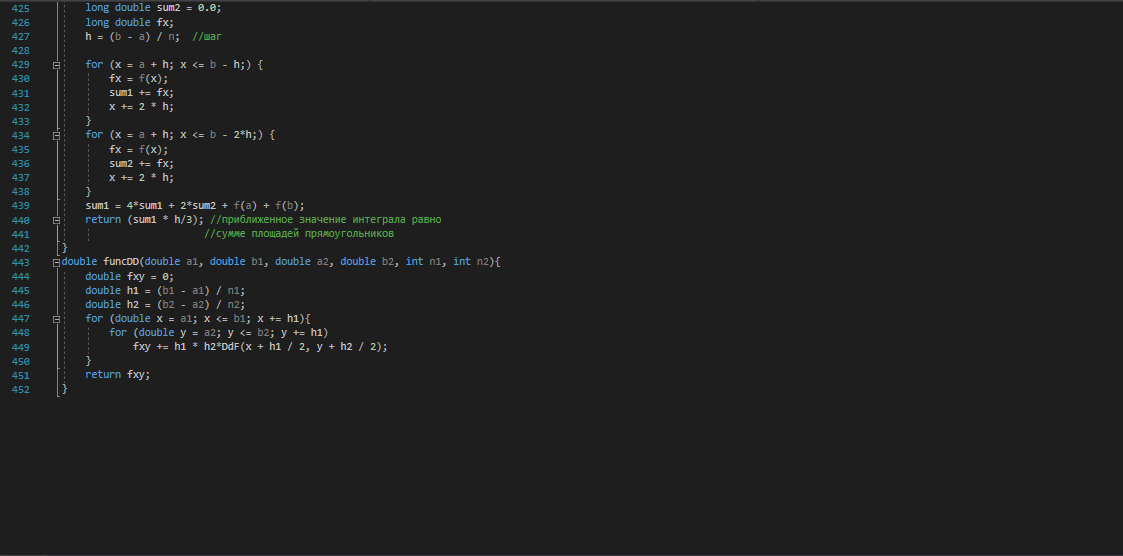






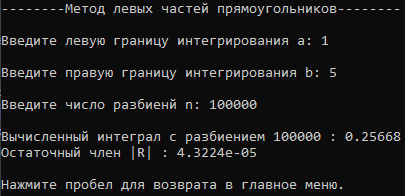


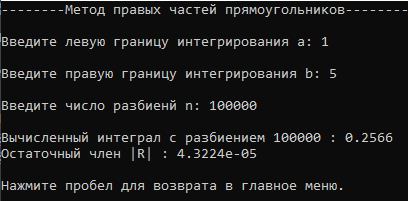


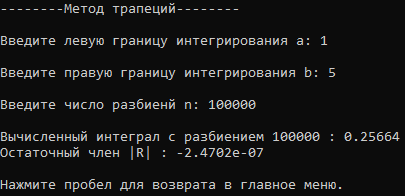


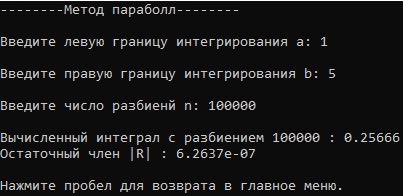
**Результаты для табличного интеграла:**

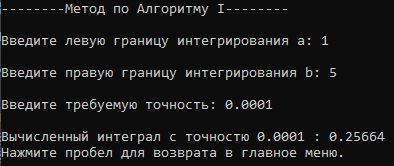
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Метод левых частей прямоугольников** | **Метод правых частей прямоугольников** | **Метод трапеций** | **Метод параболл** | **Алгоритм I** | **Алгоритм II** |
| 0.25668, R = 4.322 | 0.2566, R = 4.322 | 0.25664, R =-2.470 | 0.25666, R =-6.264 | e = 0.0001, 0.25664 | e = 0.0001, 0.25664 |

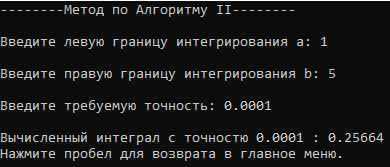






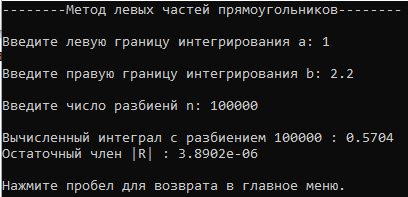


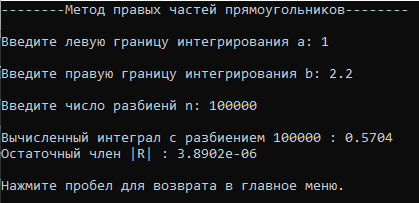


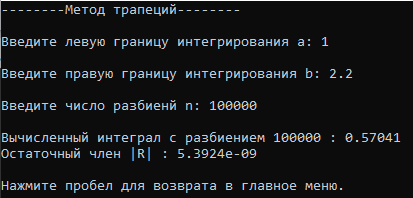


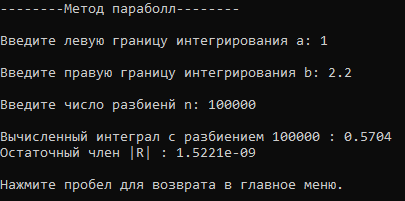
**Результаты для произвольного интеграла:**

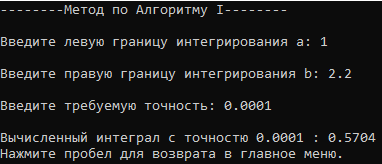
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Метод левых частей прямоугольников** | **Метод правых частей прямоугольников** | **Метод трапеций** | **Метод параболл** | **Алгоритм I** | **Алгоритм II** |
| 0.5704, R = 3.890 | 0.5704, R = 3.890 | 0.57041, R = 5.392 | 0.5704, R =1.522 | e = 0.0001, 0.5704 | e = 0.0001, 0.57059 |

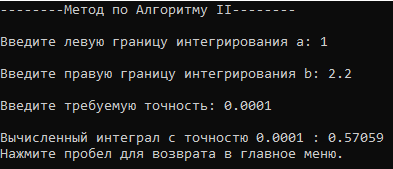




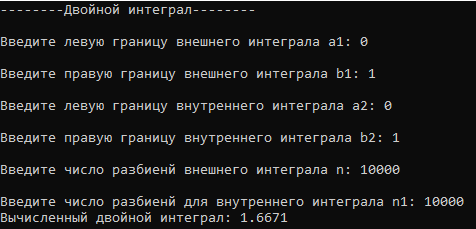








**Результат для двойного интеграла:**



**Вывод:**

Были реализованы разные методы интегрирования программным путем, можно увидеть их различия по результатам работы. Проанализировав полученные данные, установлено, что результаты вычислений зависят от выбранного метода, фукнции, количества разбиений и заданной точности, а также от самой программы.

**#include <iostream>**

**#include <cstdio>**

**#include <conio.h>**

**#include <math.h>**

**#include <Windows.h>**

**using namespace std;**

**//Прототипы функций меню**

**void MainMenu();**

**void UnderMenu1();**

**void UnderMenu2();**

**//Прототипы функций**

**typedef long double(\*pointFunc)(double);// Указатель на функцию, чтобы передавать её в качестве аргумента**

**long double f(double x); // Функция**

**double DdF(double x, double y); // Двойная функция**

**double funcDD(double a1, double b1, double a2, double b2, int n1, int n2); // Функция вычисления двойного интеграла**

**long double FirstDer(double x); // Первая производная функции**

**long double SecondDer(double x); // Вторая производная функции**

**long double ThirdDer(double x); // Третья производная функции**

**long double FourthDer(double x); // Четвертая производная функции**

**long double MaxOfDer(pointFunc f, double a, double b, int n); // Функция максимума производной**

**long double LeftRectIntegral(pointFunc f, double a, double b, int n) ; // Функция рассчета интеграла методом левых прямоугольников**

**long double RightRectIntegral(pointFunc f, double a, double b, int n); // Функция рассчета интеграла методом правых прямоугольников**

**long double TrapezIntegral(pointFunc f, double a, double b, int n); // Функция рассчета интеграла методом трапеции**

**long double TrapezIntegralAlgI(pointFunc f, double a, double b, double h); // Функция рассчета интеграла с переменным шагом методом трапеции**

**long double ParabolIntegral(pointFunc f, double a, double b, int n); // Функция рассчета интеграла методом парабол**

**int main(int argc, char\* argv[]) {**

**setlocale(NULL,"rus");**

**long double a, b, a2, b2, eps, s1, s; // Объявление нужных переменных**

**double R, h;**

**int n = 0;**

**int n1 = 0;**

**int point = -1;**

**int point2 = -1;**

**cout.precision(5); // Знаки после запятой вычислений**

**while (point != 4){ // Начало работы меню**

**R = 0.0;**

**a = 0;**

**b = 0;**

**eps = 0;**

**s1 = 0;**

**s = 0;**

**h = 0.0;**

**switch(point){**

**case -1:**

**{**

**MainMenu(); // Главное меню**

**point = -2;**

**}**

**case -2:**

**{**

**cout << "\n>> ";**

**cin >> point;**

**break;**

**}**

**case 1:**

**{**

**switch (point2) {**

**case -1:**

**{**

**system("cls");**

**UnderMenu1(); //Меню для вычисления с постоянным шагом**

**point2 = -2;**

**}**

**case -2:**

**{**

**cout << "\n>> ";**

**cin >> point2;**

**break;**

**}**

**case 1:**

**{**

**system("cls");**

**cout << "--------Метод левых частей прямоугольников--------" << endl;**

**cout << "\nВведите левую границу интегрирования a: ";**

**cin >> a;**

**cout << "\nВведите правую границу интегрирования b: ";**

**cin >> b;**

**cout << "\nВведите число разбиенй n: ";**

**cin >> n;**

**R = (pow((b-a),2) / (2 \* n)) \* MaxOfDer(FirstDer, a, b, n) ;**

**cout << "\nВычисленный интеграл c разбиением " << n << " : " << LeftRectIntegral(f, a, b, n)**

**<< "\nОстаточный член |R| : "<< R << endl;**

**cout << "\nНажмите пробел для возврата в главное меню." << endl;**

**\_getch();**

**system("cls");**

**point2 = -1;**

**point = -1;**

**break;**

**}**

**case 2:**

**{**

**system("cls");**

**cout << "--------Метод правых частей прямоугольников--------" << endl;**

**cout << "\nВведите левую границу интегрирования a: ";**

**cin >> a;**

**cout << "\nВведите правую границу интегрирования b: ";**

**cin >> b;**

**cout << "\nВведите число разбиенй n: ";**

**cin >> n;**

**R = (pow((b - a), 2) / (2 \* n)) \* MaxOfDer(FirstDer, a, b, n);**

**cout << "\nВычисленный интеграл c разбиением " << n << " : " << RightRectIntegral(f, a, b, n)**

**<< "\nОстаточный член |R| : " << R << endl;**

**cout << "\nНажмите пробел для возврата в главное меню." << endl;**

**\_getch();**

**system("cls");**

**point2 = -1;**

**point = -1;**

**break;**

**}**

**case 3:**

**{**

**system("cls");**

**cout << "--------Метод трапеций--------" << endl;**

**cout << "\nВведите левую границу интегрирования a: ";**

**cin >> a;**

**cout << "\nВведите правую границу интегрирования b: ";**

**cin >> b;**

**cout << "\nВведите число разбиенй n: ";**

**cin >> n;**

**R = (pow((b - a), 3) / (12 \* n\*n)) \* MaxOfDer(SecondDer, a, b, n);**

**cout << "\nВычисленный интеграл c разбиением " << n << " : " << TrapezIntegral(f, a, b, n)**

**<< "\nОстаточный член |R| : " << R << endl;**

**cout << "\nНажмите пробел для возврата в главное меню." << endl;**

**\_getch();**

**system("cls");**

**point2 = -1;**

**point = -1;**

**break;**

**}**

**case 4:**

**{**

**system("cls");**

**cout << "--------Метод параболл--------" << endl;**

**cout << "\nВведите левую границу интегрирования a: ";**

**cin >> a;**

**cout << "\nВведите правую границу интегрирования b: ";**

**cin >> b;**

**cout << "\nВведите число разбиенй n: ";**

**cin >> n;**

**R = (pow((b - a), 5) / (180 \* 4\*n\*n)) \* MaxOfDer(FourthDer, a, b, n);**

**s1 = ParabolIntegral(f, a, b, n); //первое приближение для интеграла**

**cout << "\nВычисленный интеграл c разбиением " << n << " : " << ParabolIntegral(f, a, b, n)**

**<< "\nОстаточный член |R| : " << R << endl;**

**cout << "\nНажмите пробел для возврата в главное меню." << endl;**

**\_getch();**

**system("cls");**

**point2 = -1;**

**point = -1;**

**break;**

**}**

**case 5:**

**{**

**point = -1;**

**point2 = -1;**

**system("cls");**

**break;**

**}**

**default:**

**{**

**cout << "Ошибка, попробуйте еще раз." << endl;**

**point2 = -2;**

**break;**

**}**

**}**

**break;**

**}**

**case 2:**

**{**

**switch (point2) {**

**case -1:**

**{**

**system("cls");**

**UnderMenu2(); //Меню для вычисления с переменным шагом**

**point2 = -2;**

**}**

**case -2:**

**{**

**cout << "\n>> ";**

**cin >> point2;**

**break;**

**}**

**case 1:**

**{**

**system("cls");**

**double h = 0.0;**

**cout << "--------Метод по Алгоритму I--------" << endl;**

**cout << "\nВведите левую границу интегрирования a: ";**

**cin >> a;**

**cout << "\nВведите правую границу интегрирования b: ";**

**cin >> b;**

**cout << "\nВведите требуемую точность: ";**

**cin >> eps;**

**h = float(sqrt(eps));**

**s1 = TrapezIntegralAlgI(f, a, b, h); //первое приближение для интеграла**

**do {**

**s = s1; //второе приближение**

**h = float(h / 2.0); //уменьшение значения шага в два раза**

**s1 = TrapezIntegralAlgI(f, a, b, h);**

**} while (abs(s1 - s) >= eps); //сравнение приближений с заданной точностью**

**cout << "\nВычисленный интеграл c точностю " << eps << " : " << s1;**

**cout << "\nНажмите пробел для возврата в главное меню." << endl;**

**\_getch();**

**system("cls");**

**point2 = -1;**

**point = -1;**

**break;**

**}**

**case 2:**

**{**

**system("cls");**

**double hv, hs, hd;**

**cout << "--------Метод по Алгоритму II--------" << endl;**

**cout << "\nВведите левую границу интегрирования a: ";**

**cin >> a;**

**cout << "\nВведите правую границу интегрирования b: ";**

**cin >> b;**

**cout << "\nВведите требуемую точность: ";**

**cin >> eps;**

**hv = float(sqrt(eps));**

**hs = float(hv / 2.0);**

**s1 = 0;**

**do {**

**s = RightRectIntegral(f, a, b, (b - a) / hv);**

**hv += hs;**

**hd = hv / 2;**

**s1 = RightRectIntegral(f, a, b, (b - a) / hd);**

**hv = float(hv / 2.0); //уменьшение значения шага в два раза**

**hs = float(hs / 2.0);**

**} while (abs(s1 - s) >= eps); //сравнение приближений с заданной точностью**

**cout << "\nВычисленный интеграл c точностю " << eps << " : " << s1;**

**cout << "\nНажмите пробел для возврата в главное меню." << endl;**

**\_getch();**

**system("cls");**

**point2 = -1;**

**point = -1;**

**break;**

**}**

**case 3:**

**{**

**point = -1;**

**point2 = -1;**

**system("cls");**

**break;**

**}**

**default:**

**{**

**cout << "Ошибка, попробуйте еще раз." << endl;**

**point2 = -2;**

**break;**

**}**

**}**

**break;**

**}**

**case 3:**

**{**

**system("cls");**

**cout << "--------Двойной интеграл--------" << endl;**

**cout << "\nВведите левую границу внешнего интеграла a1: ";**

**cin >> a;**

**cout << "\nВведите правую границу внешнего интеграла b1: ";**

**cin >> b;**

**cout << "\nВведите левую границу внутреннего интеграла a2: ";**

**cin >> a2;**

**cout << "\nВведите правую границу внутреннего интеграла b2: ";**

**cin >> b2;**

**cout << "\nВведите число разбиенй внешнего интеграла n: ";**

**cin >> n;**

**cout << "\nВведите число разбиенй для внутреннего интеграла n1: ";**

**cin >> n1;**

**cout << "Вычисленный двойной интеграл: " << funcDD(a, b, a2, b2, n, n1) << endl;**

**\_getch();**

**system("cls");**

**point2 = -1;**

**point = -1;**

**break;**

**}**

**case 4:**

**{**

**break;**

**}**

**default:**

**{**

**cout << "Ошибка, попробуйте еще раз." << endl;**

**point = -2;**

**}**

**}**

**}**

**\_getch ();**

**return 0;**

**}**

**void MainMenu() {**

**cout << "--------Главное меню--------" << endl << endl**

**<< "1) Методы с постоянным шагом\n"**

**<< "2) Методы с переменным шагом\n"**

**<< "3) Двойной интеграл\n"**

**<< "4) Выход из программы" << endl;**

**}**

**void UnderMenu1() {**

**cout << "--------Методы с постоянным шагом--------" << endl << endl**

**<< "1) Метод левых частей прямоугольников\n"**

**<< "2) Метод правых частей прямоугольников\n"**

**<< "3) Метод трапеций\n"**

**<< "4) Метод параболл\n"**

**<< "5) Возврат в главное меню" << endl;**

**}**

**void UnderMenu2() {**

**cout << "--------Методы с переменным шагом--------" << endl << endl**

**<< "1) Метод по I алгоритму\n"**

**<< "2) Метод по II алгоритму\n"**

**<< "3) Возврат в главное меню" << endl;**

**}**

**// Определение функций**

**long double f(double x) {**

**return sin(x);**

**}**

**double DdF(double x, double y){**

**return (x\*x + y\*y +1);**

**}**

**long double FirstDer(double x) {**

**return cos(x);**

**}**

**long double SecondDer(double x) {**

**return -sin(x);**

**}**

**long double ThirdDer(double x) {**

**return -cos(x);**

**}**

**long double FourthDer(double x) {**

**return sin(x);**

**}**

**long double MaxOfDer(pointFunc f, double a, double b, int n) {**

**double h;**

**double max;**

**h = (b - a) / n;**

**max = f(a);**

**for (float i = a + h; i <= b; i += h) {**

**if (max < f(i))**

**max = f(i);**

**}**

**return max;**

**}**

**long double LeftRectIntegral(pointFunc f, double a, double b, int n) {**

**double x, h;**

**long double sum = 0.0;**

**h = float((b - a) / n); //шаг**

**for (x = a; x <= b - h;) {**

**sum += f(x);**

**x += h;**

**}**

**return (sum \* h); //приближенное значение интеграла равно**

**//сумме площадей прямоугольников**

**}**

**long double RightRectIntegral(pointFunc f, double a, double b, int n) {**

**double x, h;**

**long double sum = 0.0;**

**h = float((b - a) / n); //шаг**

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

**sum += f(x);**

**x += h;**

**}**

**return (sum \* h); //приближенное значение интеграла равно**

**//сумме площадей прямоугольников**

**}**

**long double TrapezIntegral(pointFunc f, double a, double b, int n) {**

**double x, h;**

**long double sum = 0.0;**

**long double fx;**

**h = (b - a) / n; //шаг**

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

**// x = a + h + i \* h;**

**fx = f(x);**

**sum += fx;**

**x += h;**

**}**

**sum += (f(a) + f(b)) / 2;**

**return (sum \* h); //приближенное значение интеграла равно**

**//сумме площадей прямоугольников**

**}**

**long double TrapezIntegralAlgI(pointFunc f, double a, double b, double h) {**

**double x;**

**long double sum = 0.0;**

**long double fx;**

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

**fx = f(x);**

**sum += fx;**

**x += h;**

**}**

**sum += (f(a) + f(b)) / 2;**

**return (sum \* h); //приближенное значение интеграла равно**

**//сумме площадей прямоугольников**

**}**

**long double ParabolIntegral(pointFunc f, double a, double b, int n) {**

**double x, h;**

**long double sum1 = 0.0;**

**long double sum2 = 0.0;**

**long double fx;**

**h = (b - a) / n; //шаг**

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

**fx = f(x);**

**sum1 += fx;**

**x += 2 \* h;**

**}**

**for (x = a + h; x <= b - 2\*h;) {**

**fx = f(x);**

**sum2 += fx;**

**x += 2 \* h;**

**}**

**sum1 = 4\*sum1 + 2\*sum2 + f(a) + f(b);**

**return (sum1 \* h/3); //приближенное значение интеграла равно**

**//сумме площадей прямоугольников**

**}**

**double funcDD(double a1, double b1, double a2, double b2, int n1, int n2){**

**double fxy = 0;**

**double h1 = (b1 - a1) / n1;**

**double h2 = (b2 - a2) / n2;**

**for (double x = a1; x <= b1; x += h1){**

**for (double y = a2; y <= b2; y += h1)**

**fxy += h1 \* h2\*DdF(x + h1 / 2, y + h2 / 2);**

**}**

**return fxy;**

**}**