**Лабораторная работа №1**

**Колбин Павел**

**Вариант 10**

Создайте пустой проект C++ в VS. Добавьте, напишите, отладьте исходные коды для расчета интеграла по Вашему варианту. Оцените время и точность (относительно аналитического значения) расчета интеграла в зависимости от количества интервалов (равномерное разбиение, 100, 1000, 10000, 100000, 1000000).

|  |
| --- |
| #include <iostream>  #include <math.h>  #include <iomanip>  #include <cmath>  #include <chrono>  **using** **namespace** std;  **using** **namespace** std::chrono;  **typedef** **double**(\*pointFunc)(**double**);  **double** f (**double** x) {  **return** (4/pow((1 + x \* x) , 2));  }  **double** simpson\_integral(pointFunc f, **double** a, **double** b, **int** n) {  **const** **double** h = (b - a) / n;  **double** k1 = 0, k2 = 0;  **for** (**int** i = 1; i < n; i += 2) {  k1 += f(a + i \* h);  k2 += f(a + (i + 1) \* h);  }  **return** h / 3 \* (f(a) + 4 \* k1 + 2 \* k2);  }  **int** main() {    **double** a = -1, b = 1, eps;  **double** s1, s;  **int** n = 1;  **int** i\_nums; //начальное число шагов  cout << "Input number of intervals = ";  cin >> i\_nums;  eps = 1.0 / i\_nums;  high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();  s1 = simpson\_integral(f, a, b, n); //первое приближение для интеграла  **do**{  s = s1; //второе приближение  n = 2 \* n; //увеличение числа шагов в два раза,  //т.е. уменьшение значения шага в два раза  s1 = simpson\_integral(f, a, b, n);  } **while** (fabs(s1 - s) > eps);  high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();  duration<**double**> duration = (t2 - t1);  cout << "Duration is: " << duration.count() << " seconds" << endl;  cout << "Integral = " << setprecision(10) << s1 << endl;  } |

|  |  |  |  |
| --- | --- | --- | --- |
| *Аналитическое решение* | *Необходимое количество разбиений* | *Численное решение* | *Затраченное время, c* |
| 5.141592653589793 | *100* | *5.146800983* | *1.28 \* 10^-5* |
| *1000* | *5.142243695* | *1.003 \* 10^-4* |
| *10000* | *5.141674034* | *7.681 \* 10^-4* |
| *100000* | *5.14159774* | *1.03851 \* 10^-2* |
| *1000000* | *5.141593289* | *7.37916 \* 10^-2* |