**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное**

**учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»**

Факультет компьютерных наук

Департамент программной инженерии

# Отчет к домашнему заданию по дисциплине

**«Архитектура вычислительных систем»**

Работу выполнил:

Студент группы БПИ195 Цуркан Даниел

**Москва 2020**

**Задание (Вариант - 29)**

Вычислить интеграл используя метод прямоугольников. Входные данные: вещественные числа a и b, функция f(x) задается c использованием описания в программе в виде отдельной функции. При суммировании использовать принцип дихотомии. Протестировать на различных функциях.

**Решение**

В начале программа просит пользователя ввести два числа: нижний и верхний предел интегрирования. При обнаружении неправильного ввода или других исключений программа просит пользователя заново ввести значения. Далее программа передает значения в метод, в котором происходит распараллеливание задач и как возвращаемое значение возвращает результат интегрирования.

Решение задачи было разбито на несколько функций: f(double x), get\_A(), get\_B(double a), meth\_for\_one\_thread(double a, int number\_of\_thread, int n, double h), get\_result(double a, double b, double h).

Функция f(double x) задает вычисляемую функцию.

Функция get\_A() просит пользователя ввести нижний предел интегрирования и при возникновении каких-либо ошибок ввода просит пользователя ввести значение заново.

Функция get\_B(double a) просит пользователя ввести нижний предел интегрирования и при возникновении каких-либо ошибок ввода просит пользователя ввести значение заново.

Функция meth\_for\_one\_thread(double a, int number\_of\_thread, int n, double h) вычисляет интеграл некоторых прямоугольников для одного потока и возвращает вычисленное значение одного потока.

Функция get\_result(double a, double b, double h) распараллеливает задачи для всех потоков, после чего складывает все и возвращает результат интегрирования.

**Текст программы**

#include <iostream>  
#include <omp.h>  
#include <vector>  
  
  
int N = omp\_get\_max\_threads();  
  
double f(double x) {  
 return x \* x\*x + 4;  
}  
  
double get\_A() {  
 std::string a1;  
 double a;  
 std::cout << "To calculate integral of function - f(x) = x^2 + 1 please input A - ";  
 std::cin >> a1;  
 try {  
 a = std::stod(a1);  
 } catch (std::exception) {  
 std::cout << "Incorrect input, try again!\n";  
 a = get\_A();  
 }  
 return a;  
}  
  
double get\_B(double a) {  
 std::string b1;  
 double b;  
 std::cout << "Please input B - ";  
 std::cin >> b1;  
 try {  
 b = std::stod(b1);  
 } catch (std::exception) {  
 std::cout << "Incorrect input, try again!\n";  
 b = get\_B(a);  
 }  
 if (b < a) {  
 std::cout << "Incorrect input, B must be more than A. Try again!\n";  
 b = get\_B(a);  
 }  
 return b;  
}  
  
// number\_thread == i  
// n - кол-во прямоугольников  
double meth\_for\_one\_thread(double a, int number\_thread, int n, double h) {  
 double thread\_sum = 0;  
 for (int i = number\_thread; i <= n; i += N) {  
 thread\_sum += f(a + i \* h);  
 }  
  
 return thread\_sum \* h;  
}  
  
double get\_result(double a, double b, double h) {  
 double n = (b - a) / h; // кол-во разбиений на прямоугольники  
 double \*thread\_res = new double[N]{0.0};  
  
#pragma omp parallel for  
 for (int i = 0; i < N; ++i) {  
 thread\_res[i] = meth\_for\_one\_thread(a, i, n, h);  
 }  
  
 double result = 0;  
 for (int j = 0; j < N; ++j) {  
 result += thread\_res[j];  
 }  
  
 delete [] thread\_res;  
 return result;  
}  
  
  
int main() {  
  
 double a = get\_A(); // Нижний предел.  
 double b = get\_B(a); // Верхний предел.  
 double h = 0.0001; // Переменная необходимая для разбиения графика на прямоугольники.  
  
 std::cout << get\_result(a, b, h);  
  
 return 0;  
}

**Тестирование**

Результат выполнения программы приведен на рисунке 1 и на рисунке 2.

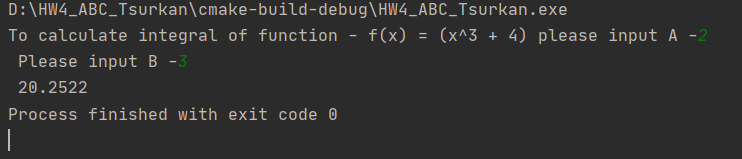


Рисунок 1 – Результат выполнения программы

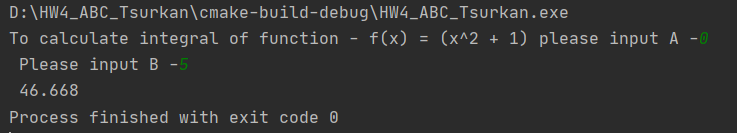


Рисунок 2 – Результат выполнения программы

**Список используемых источников**

1. SoftCraft разноликое программирование «Программирование на языке ассемблера. Микропроект. Требования к оформлению. 2020-2021 уч.г.» (<http://softcraft.ru/edu/comparch/tasks/mp01/>)
2. https://docs.microsoft.com/ru-ru/cpp/parallel/openmp/reference/openmp-directives?view=msvc-160