

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра математической кибернетики и компьютерных наук

ОТЧЕТ ПО ПРАКТИКЕ WORK10

ОТЧЕТ

Студента 3 курса 311 группы
направления 02.03.02 — Фундаментальная информатика и информационные
технологии
факультета КНиИТ
Забоева Максима Владиславовича

Проверил

Старший преподаватель

М. С. Портенко

Саратов 2023

СОДЕРЖАНИЕ

1	Условие задачи	3
2	Практическая часть	4
2.1	Метод прямоугольников	4
2.2	Формула Симпсона	5
3	Результаты работы	6
3.1	Характеристики компьютера	6
3.2	Фото результатов	6

1 Условие задачи

Аналогично работе с ОМР выполните следующее задание через MPI.

Реализуйте параллельные алгоритмы, использующие метод прямоугольников и формулу Симпсона для подсчета интегралов. Точные значения интегралов указаны для проверки численных вычислений. В случае, если в верхнем пределе интегрирования указан знак бесконечности, то в расчете необходимо заменить его на 10^6

Вариант 6

2 Практическая часть

2.1 Метод прямоугольников

Код программы:

```
#include <iostream>
#include <mpi.h>
#include <time.h>
#include <cmath>
using namespace std;
double f(double x) {
    return ((1.0) / (x * x + 4 * x + 5));
}
double integral(const double a, const double b, const double h) {
    int i, n;
    double sum, res = 0; // локальная переменная для подсчета интеграла
    double x; // координата точки сетки
    n = (int)((b - a) / h); // количество точек сетки интегрирования
    sum = 0.0;
    int commsize;
    int rank;
    double Result;
    MPI_Init(NULL, NULL);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &commsize);
    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
    for (i = 0; i < n; i++)
    {
        x = a + i * h + h / 2.0;
        sum += f(x) * h;
    }
    // (in, out, count, type, op, номер главного процесса, коммуникатор)
    MPI_Reduce(&sum, &Result, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
    MPI_Finalize();
    return Result;
}
int main()
{
    double min_time; // минимальное время работы
    // реализации алгоритма
    double max_time; // максимальное время работы
    // реализации алгоритма
    double avg_time; // среднее время работы
    double res;
    double a = -1.0;
    double b = 1000000;
    double h = 0.1;
    int n = 6;
    min_time = clock();
    res = integral(a, b, h);
    max_time = clock();
    avg_time = (max_time - min_time) / CLOCKS_PER_SEC;
    cout << "execution time : " << avg_time << endl;
    cout.precision(8);
    cout << "integral value : " << res << endl;
    return 0;
}
```

2.2 Формула Симпсона

Код программы:

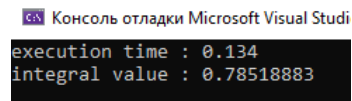
```
#include <iostream>
#include <mpi.h>
#include <time.h>
#include <cmath>
using namespace std;
double f(double x) {
    return ((1.0) / (x * x + 4 * x + 5));
}
double integral(const double a, const double b, const double h) {
    int k, n;
    double sum;
    double sum1 = 0.0;
    double sum2 = 0.0;
    n = (int)((b - a) / h);
    sum = 0.0;
    int commsize;
    int rank;
    double Result1, Result2;
    MPI_Init(NULL, NULL);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &commsize);
    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
    for (k = 1; k <= n; k++) {
        double sum_f_1 = a + ((2 * k - 1) * h);
        double sum_f_2 = a + (2 * k * h);
        sum1 += f(sum_f_1);
        if (k < n)
            sum2 += f(sum_f_2);
    }
    MPI_Reduce(&sum1, &Result1, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
    MPI_Reduce(&sum2, &Result2, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
    MPI_Finalize();
    sum = (h / 3.0) * ((f(a) + f(b) + (4.0) * sum1 + (2.0) * sum2));
    return sum;
}
int main()
{
    double min_time; // минимальное время работы
    // реализации алгоритма
    double max_time; // максимальное время работы
    // реализации алгоритма
    double avg_time; // среднее время работы
    double res;
    double a = -1.0;
    double b = 1000000;
    double h = 0.1;
    int n = 6;
    min_time = clock();
    res = integral(a, b, h);
    max_time = clock();
    avg_time = (max_time - min_time) / CLOCKS_PER_SEC;
    cout << "execution time : " << avg_time << endl;
    cout.precision(8);
    cout << "integral value : " << res << endl;
    return 0;
}
```

3 Результаты работы

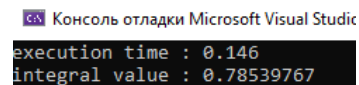
3.1 Характеристики компьютера

Процессор — 12th Gen Intel Core i5-12600KF, Базовая скорость 3,70ГГц,
Кол-во ядер 10, Кол-во процессоров 16 (включая 4 энергоэффективных ядра).
16гб Оперативной памяти, скорость 3200МГц

3.2 Фото результатов



Консоль отладки Microsoft Visual Studi
execution time : 0.134
integral value : 0.78518883



Консоль отладки Microsoft Visual Studi
execution time : 0.146
integral value : 0.78539767