

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

**ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ, ФОРМУЛА СИМПСОНА,  
ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ И КРАТНЫХ ИНТЕГРАЛОВ**

**ОТЧЕТ О ПРАКТИКЕ**

Студента 3 курса 311 группы  
направления 02.03.02 — Фундаментальная информатика и информационные  
технологии  
факультета КНиИТ  
Аношкина Андрея Алексеевича

Проверил

Старший преподаватель

\_\_\_\_\_

М. С. Портенко

Саратов 2024

## СОДЕРЖАНИЕ

1	Work 03.....	3
---	--------------	---

## 1 Work 03

### Задание

Реализовать параллельно один из методов приближенного вычисления двойных интегралов, а именно метод статистических испытаний.

### Метод трапеций

Пусть и по направлению  $x$ , и по направлению  $y$  для приближенного вычисления применяется формула трапеций.

$$F(y_j) \approx h_1 * \sum_{i=0}^m q_{1,i} * f(x_i, y_j)$$

где  $q_{1,i} = 0.5$ , при  $i = 0$  и  $i = m$ ;  $q_{1,i} = 1$ , при  $i = 1, 2, \dots, m - 1$   
и

$$J \approx h_2 * \sum_{j=0}^n q_{2,j} * F(y_j)$$

где  $q_{2,j} = 0.5$ , при  $j = 0$  и  $j = n$ ;  $q_{2,j} = 1$ , при  $j = 1, 2, \dots, n - 1$   
Тогда

$$J \approx h_1 * h_2 * \sum_{i=0}^m \sum_{j=0}^n q_{ij} * f(x_i, y_j)$$

где  $q_{ij} = q_{1,i} * q_{2,j}$

### Реализация

Фрагмент кода решения приведен ниже:

```
1  #include <iostream>
2  #include <omp.h>
3  #include <time.h>
4  #include <cmath>
5  #define PI 3.1415926535897932384626433832795
6
7  using namespace std;
8
9  double f1(const double x, const double y, const double a1, const double b1, const double a2, const double b2) {
10     return (1 + exp(sin(PI * x) * cos(PI * y))) / (b1 - a1) / (b2 - a2);
11 }
12
13 double q(int i, int j, int n, int m) {
14     double q = 1;
```

```

15     if (i == 0 || i == n)
16         q *= 0.5;
17     if (j == 0 || j == m)
18         q *= 0.5;
19     return q;
20 }
21
22 void integral_posl(const double a1, const double b1, const double a2, const double b2, const double h1, const
↪ double h2, double* res) {
23     double sum = 0;
24     int n = (int)((b1 - a1) / h1);
25     int m = (int)((b2 - a2) / h2);
26
27     for (int i = 0; i <= n; ++i) {
28         double x = a1 + i * h1 + h1 / 2;
29         for (int j = 0; j <= m; ++j) {
30             double y = a2 + j * h2 + h2 / 2;
31             sum += f1(x, y, a1, b1, a2, b2) * h1 * h2 * q(i, j, n, m);
32         }
33     }
34
35     *res = sum;
36 }
37
38 void integral_parallel(const double a1, const double b1, const double a2, const double b2, const double h1, const
↪ double h2, double* res) {
39     double sum = 0;
40     int n = (int)((b1 - a1) / h1);
41     int m = (int)((b2 - a2) / h2);
42
43     #pragma omp parallel for reduction(+: sum)
44     for (int i = 0; i <= n; ++i) {
45         double x = a1 + i * h1 + h1 / 2;
46         for (int j = 0; j <= m; ++j) {
47             double y = a2 + j * h2 + h2 / 2;
48             sum += f1(x, y, a1, b1, a2, b2) * h1 * h2 * q(i, j, n, m);
49         }
50     }
51
52     *res = sum;
53 }
54
55 double experiment(double* res, void f(const double a1, const double b1, const double a2, const double b2,
↪ const double h1, const double h2, double* res)) {
56     double stime = 0, ftime = 0;
57     double a1 = 0, b1 = 16;
58     double a2 = 0, b2 = 16;
59     double h1 = 0.005, h2 = 0.005;
60     stime = clock();
61     f(a1, b1, a2, b2, h1, h2, res);

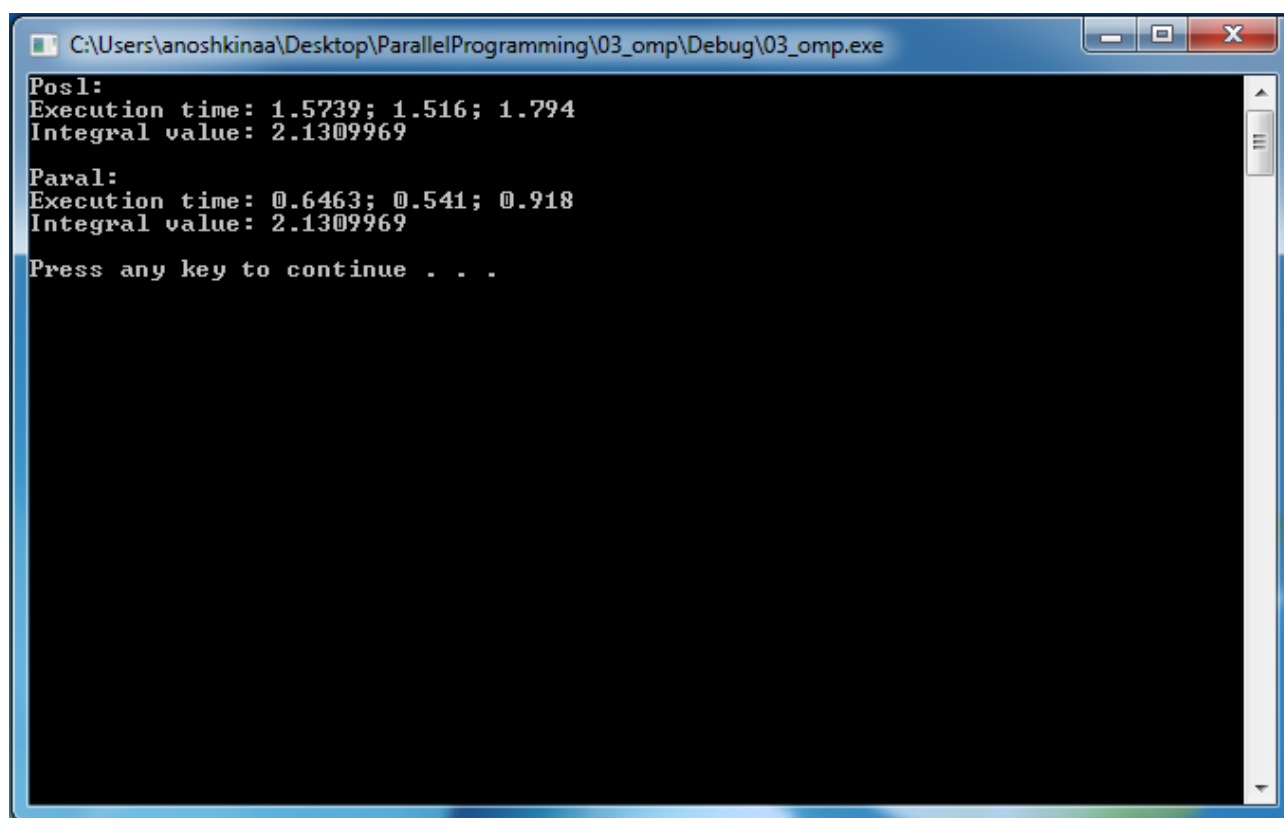
```

```

62     ftime = clock();
63
64     return (ftime - stime) / CLOCKS_PER_SEC;
65 }
66
67 void calculate(void f(const double a1, const double b1, const double a2, const double b2, const double h1,
↪ const double h2, double* res)) {
68     double avg_time = 0;
69     double min_time = 0;
70     double max_time = 0;
71     double res = 0;
72     int numbExp = 10;
73
74     min_time = max_time = experiment(&res, f);
75     avg_time = min_time / numbExp;
76     for (int i = 0; i < numbExp - 1; ++i) {
77         double time = experiment(&res, f);
78         if (time > max_time)
79             max_time = time;
80         if (time < min_time)
81             min_time = time;
82
83         avg_time += time / numbExp;
84     }
85
86     cout << "Execution time: " << avg_time << "; " << min_time << "; " << max_time << "\n";
87     cout.precision(8);
88     cout << "Integral value: " << res << "\n";
89 }
90
91 int main() {
92
93
94     cout << "Posl:\n";
95     calculate(integral_posl);
96     cout << "\n";
97
98     cout << "Paral:\n";
99     calculate(integral_paral);
100    cout << "\n";
101
102    system("pause");
103    return 0;
104 }

```

## Результат работы



```
C:\Users\anoshkinaa\Desktop\ParallelProgramming\03_omp\Debug\03_omp.exe
Pos1:
Execution time: 1.5739; 1.516; 1.794
Integral value: 2.1309969

Paral:
Execution time: 0.6463; 0.541; 0.918
Integral value: 2.1309969

Press any key to continue . . .
```

Рисунок 1 – Work-3

Значение ускорения выполнения программы при переходе к многопоточной версии:

$$a = \frac{timeSeq}{timePar} = \frac{1.5739}{0.6463} \approx 2.4352$$

### Характеристики устройства

Процессор: Intel(R) Core(TM) i3

Ядер: 4

Оперативная память: 4 Гб