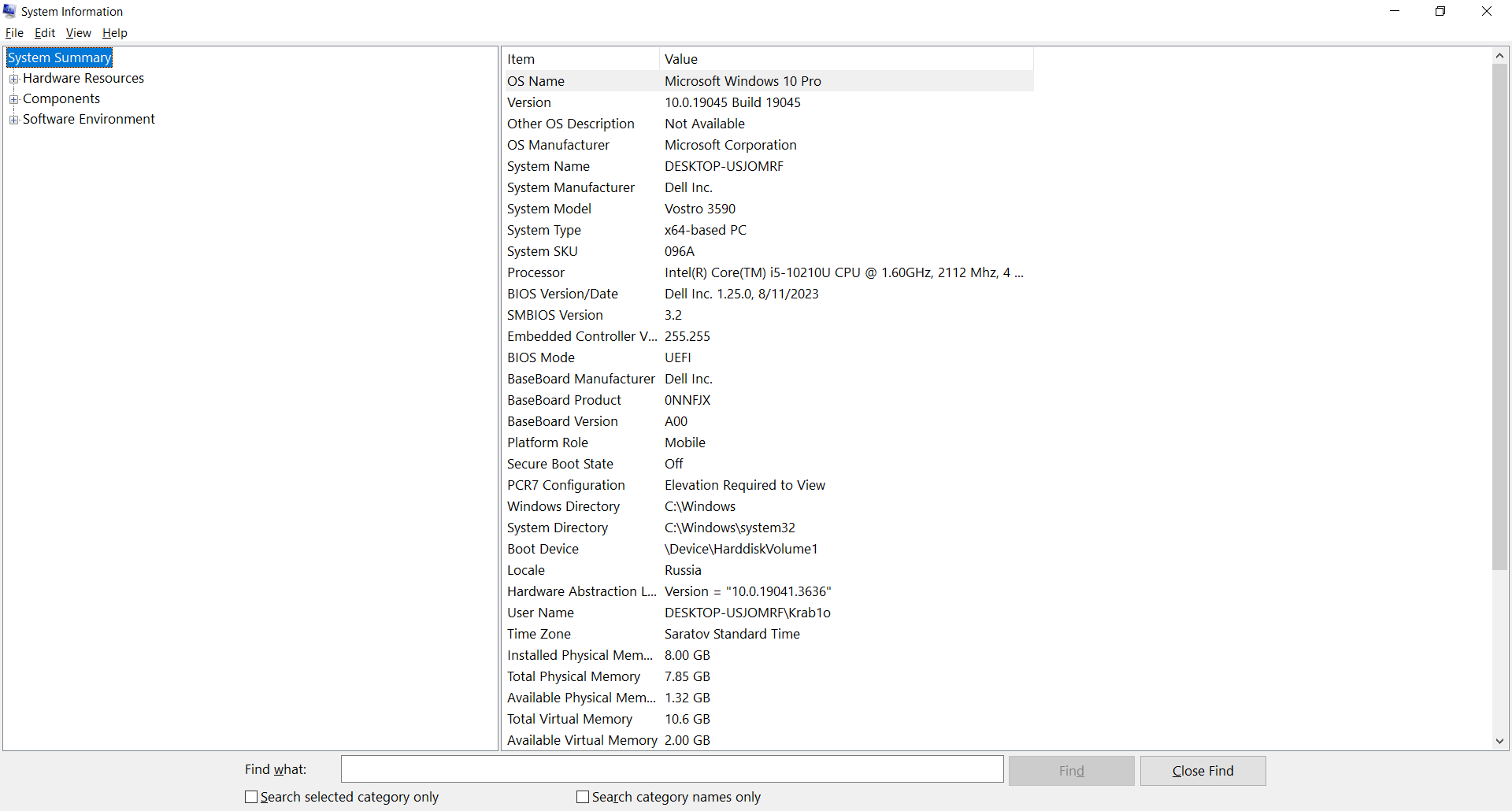
Задание выполнялось на ноутбуке со следующими параметрами:

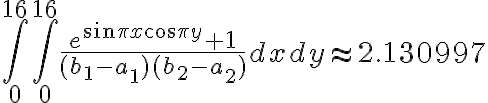


Из важного:

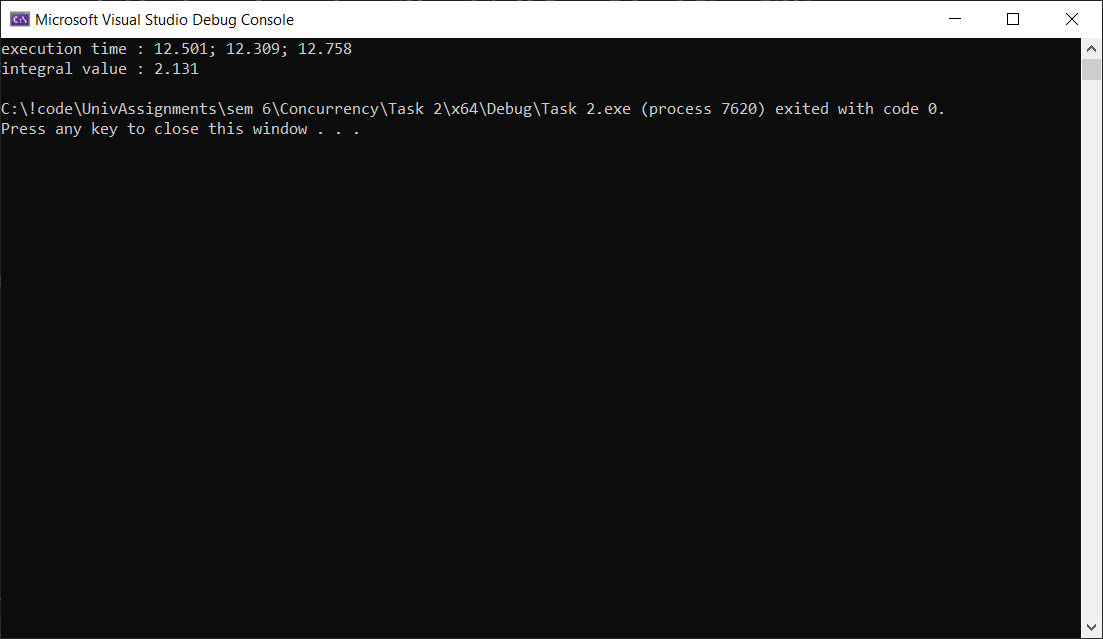
CPU: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz, 2112 Mhz, 4 Core(s), 8 Logical Processor(s)

RAM: 8.00 GB

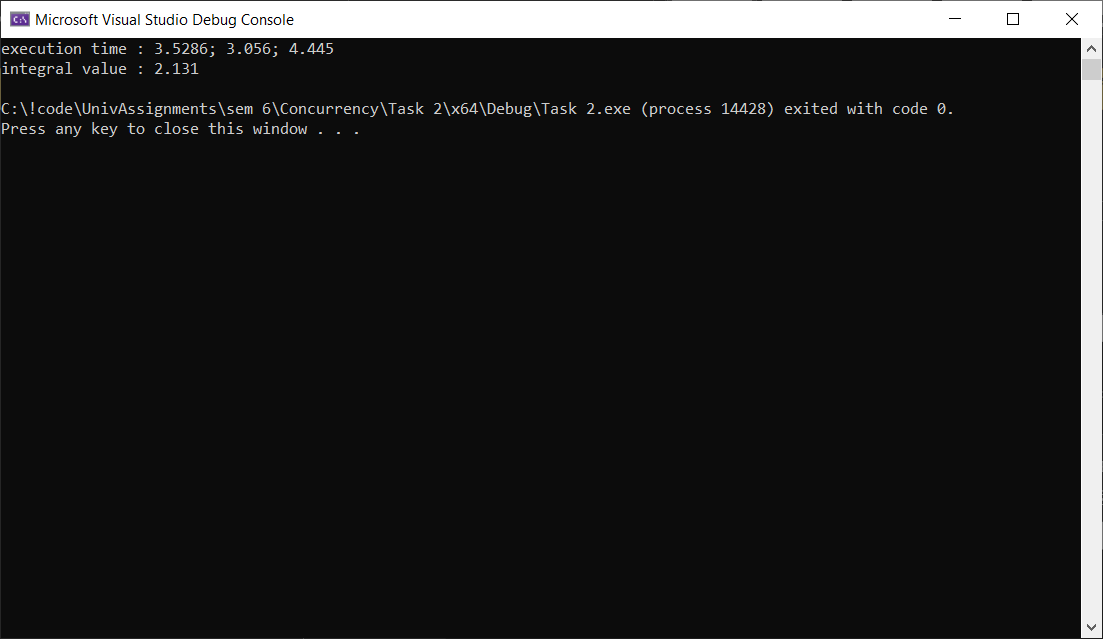
Задание. Реализовать подсчёт следующего интеграла:



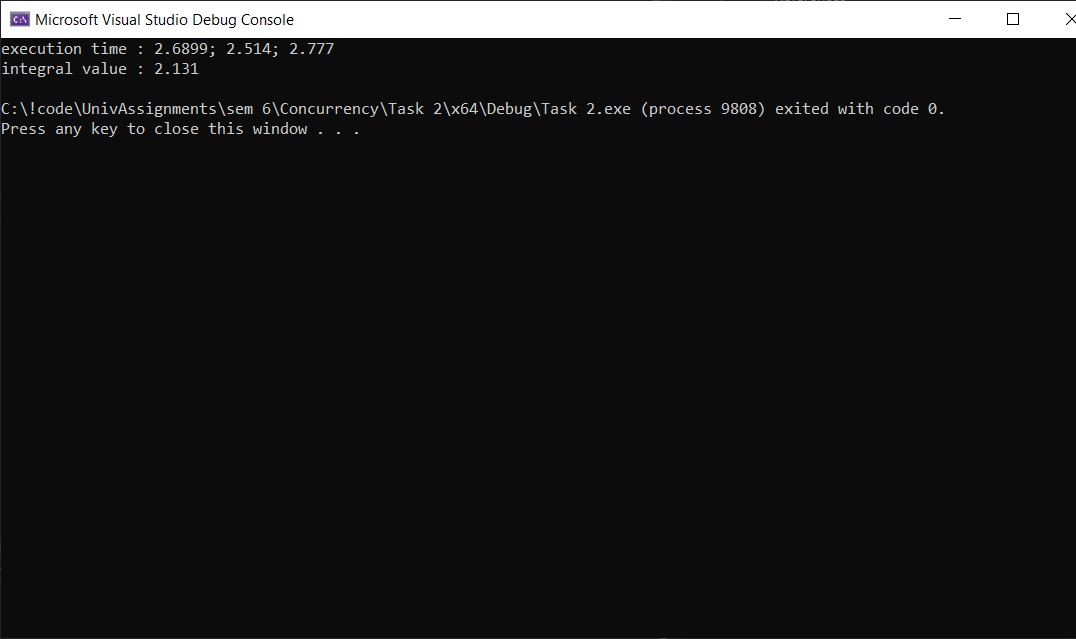
Сначала посчитаем интеграл без применения распараллеливания:



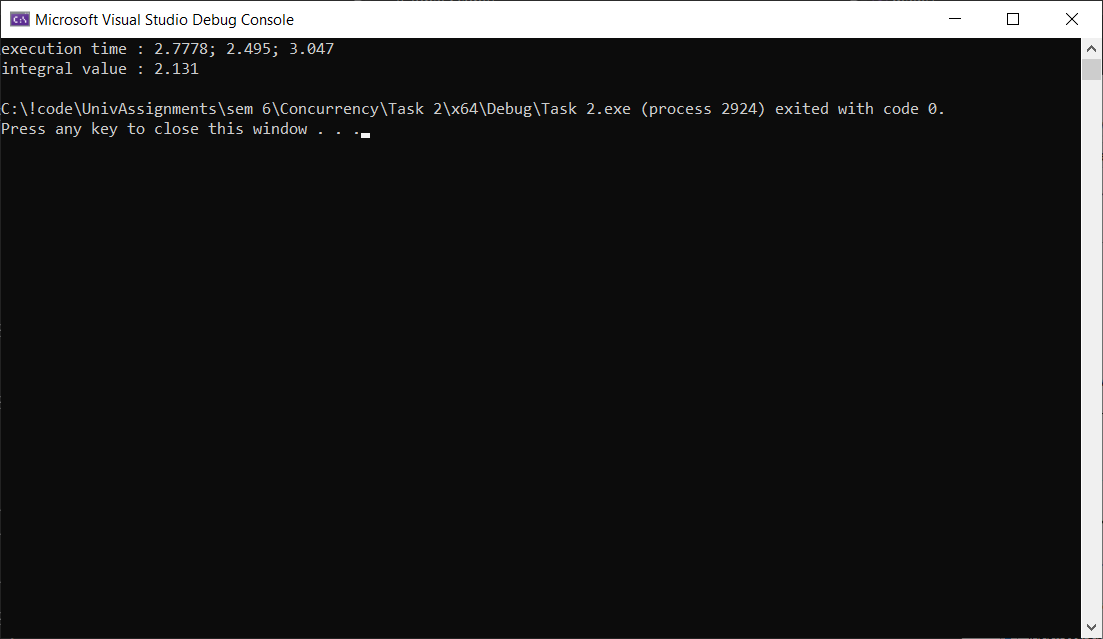
Совершим вычисления 10 раз, посчитаем среднее. В данном случае использовалось распараллеливание только внутреннего цикла.



Совершим вычисления 10 раз, посчитаем среднее. Во втором случае использовалось распараллеливание как внутреннего, так и внешнего циклов.



Только внешний цикл:



**Код для алгоритма без распараллеливания:**

#include <iostream>

#include <omp.h>

#include <vector>

#include <cmath>

#define PI 3.1415926535897932384626433832795

#define INF 1'000'000

using namespace std;

double f(double x, double y, const double& a\_x, const double& b\_x, const double& a\_y, const double& b\_y)

{

return (exp(sin(PI \* x) \* cos(PI \* y)) + 1) / ((b\_x - a\_x) \* (b\_y - a\_y));

}

void integral(const double a\_x, const double b\_x, const double a\_y, const double b\_y,

const double h\_x, const double h\_y, double\* res)

{

long long int i\_x, i\_y;

double x, y;

const long long int n\_x = (long long int)((b\_x - a\_x) / h\_x);

const long long int n\_y = (long long int)((b\_y - a\_y) / h\_y);

double sum = 0.0;

for (i\_x = 0; i\_x < n\_x; i\_x++)

{

x = a\_x + i\_x \* h\_x + h\_x / 2.0;

for (i\_y = 0; i\_y < n\_y; i\_y++)

{

y = a\_y + i\_y \* h\_y + h\_y / 2.0;

sum += f(x, y, a\_x, b\_x, a\_y, b\_y) \* h\_x \* h\_y;

}

}

\*res = sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a\_x = 0; // левая граница интегрирования по x

double b\_x = 16; // правая граница интегрирования по y

double a\_y = 0; // левая граница интегрирования по x

double b\_y = 16; // правая граница интегрирования по y

//long long int n = 1'000'000'000;

double h\_x = 0.001; // шаг интегрирования по x

double h\_y = 0.001; // шаг интегрирования по y

stime = clock();

integral(a\_x, b\_x, a\_y, b\_y, h\_x, h\_y, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time;

double max\_time;

double avg\_time;

int numbExp = 10;

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << "execution time : " << avg\_time / numbExp << "; " <<

min\_time << "; " << max\_time << endl;

cout << "integral value : " << res << endl;

return 0;

}

**Код для алгоритма с распараллеливанием внутреннего цикла:**

#include <iostream>

#include <omp.h>

#include <vector>

#include <cmath>

#define PI 3.1415926535897932384626433832795

#define INF 1'000'000

using namespace std;

double f(double x, double y, const double& a\_x, const double& b\_x, const double& a\_y, const double& b\_y)

{

return (exp(sin(PI \* x) \* cos(PI \* y)) + 1) / ((b\_x - a\_x) \* (b\_y - a\_y));

}

void integral(const double a\_x, const double b\_x, const double a\_y, const double b\_y,

const double h\_x, const double h\_y, double\* res)

{

long long int i\_x, i\_y;

double x, y;

const long long int n\_x = (long long int)((b\_x - a\_x) / h\_x);

const long long int n\_y = (long long int)((b\_y - a\_y) / h\_y);

double sum = 0.0;

for (i\_x = 0; i\_x < n\_x; i\_x++)

{

x = a\_x + i\_x \* h\_x + h\_x / 2.0;

#pragma omp parallel for private(i\_y, y) reduction(+: sum)

for (i\_y = 0; i\_y < n\_y; i\_y++)

{

y = a\_y + i\_y \* h\_y + h\_y / 2.0;

sum += f(x, y, a\_x, b\_x, a\_y, b\_y) \* h\_x \* h\_y;

}

}

\*res = sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a\_x = 0; // левая граница интегрирования по x

double b\_x = 16; // правая граница интегрирования по y

double a\_y = 0; // левая граница интегрирования по x

double b\_y = 16; // правая граница интегрирования по y

//long long int n = 1'000'000'000;

double h\_x = 0.001; // шаг интегрирования по x

double h\_y = 0.001; // шаг интегрирования по y

stime = clock();

integral(a\_x, b\_x, a\_y, b\_y, h\_x, h\_y, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time;

double max\_time;

double avg\_time;

int numbExp = 10;

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << "execution time : " << avg\_time / numbExp << "; " <<

min\_time << "; " << max\_time << endl;

cout << "integral value : " << res << endl;

return 0;

}

**Код для алгоритма с распараллеливание обоих циклов:**

v#include <iostream>

#include <omp.h>

#include <vector>

#include <cmath>

#define PI 3.1415926535897932384626433832795

#define INF 1'000'000

using namespace std;

double f(double x, double y, const double& a\_x, const double& b\_x, const double& a\_y, const double& b\_y)

{

return (exp(sin(PI \* x) \* cos(PI \* y)) + 1) / ((b\_x - a\_x) \* (b\_y - a\_y));

}

void integral(const double a\_x, const double b\_x, const double a\_y, const double b\_y,

const double h\_x, const double h\_y, double\* res)

{

long long int i\_x, i\_y;

double x, y;

const long long int n\_x = (long long int)((b\_x - a\_x) / h\_x);

const long long int n\_y = (long long int)((b\_y - a\_y) / h\_y);

double sum = 0.0;

#pragma omp parallel for private(i\_x, x) reduction (+: sum)

for (i\_x = 0; i\_x < n\_x; i\_x++)

{

x = a\_x + i\_x \* h\_x + h\_x / 2.0;

#pragma omp parallel for private (i\_y, y)

for (i\_y = 0; i\_y < n\_y; i\_y++)

{

y = a\_y + i\_y \* h\_y + h\_y / 2.0;

sum += f(x, y, a\_x, b\_x, a\_y, b\_y) \* h\_x \* h\_y;

}

}

\*res = sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a\_x = 0; // левая граница интегрирования по x

double b\_x = 16; // правая граница интегрирования по y

double a\_y = 0; // левая граница интегрирования по x

double b\_y = 16; // правая граница интегрирования по y

//long long int n = 1'000'000'000;

double h\_x = 0.001; // шаг интегрирования по x

double h\_y = 0.001; // шаг интегрирования по y

stime = clock();

integral(a\_x, b\_x, a\_y, b\_y, h\_x, h\_y, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time;

double max\_time;

double avg\_time;

int numbExp = 10;

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << "execution time : " << avg\_time / numbExp << "; " <<

min\_time << "; " << max\_time << endl;

cout << "integral value : " << res << endl;

return 0;

}

**Код для алгоритма с распараллеливанием внешнего цикла:**

#include <iostream>

#include <omp.h>

#include <vector>

#include <cmath>

#define PI 3.1415926535897932384626433832795

#define INF 1'000'000

using namespace std;

double f(double x, double y, const double& a\_x, const double& b\_x, const double& a\_y, const double& b\_y)

{

return (exp(sin(PI \* x) \* cos(PI \* y)) + 1) / ((b\_x - a\_x) \* (b\_y - a\_y));

}

void integral(const double a\_x, const double b\_x, const double a\_y, const double b\_y,

const double h\_x, const double h\_y, double\* res)

{

long long int i\_x, i\_y;

double x, y;

const long long int n\_x = (long long int)((b\_x - a\_x) / h\_x);

const long long int n\_y = (long long int)((b\_y - a\_y) / h\_y);

double sum = 0.0;

#pragma omp parallel for private(i\_x, x, i\_y, y) reduction (+: sum)

for (i\_x = 0; i\_x < n\_x; i\_x++)

{

x = a\_x + i\_x \* h\_x + h\_x / 2.0;

for (i\_y = 0; i\_y < n\_y; i\_y++)

{

y = a\_y + i\_y \* h\_y + h\_y / 2.0;

sum += f(x, y, a\_x, b\_x, a\_y, b\_y) \* h\_x \* h\_y;

}

}

\*res = sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a\_x = 0; // левая граница интегрирования по x

double b\_x = 16; // правая граница интегрирования по y

double a\_y = 0; // левая граница интегрирования по x

double b\_y = 16; // правая граница интегрирования по y

//long long int n = 1'000'000'000;

double h\_x = 0.001; // шаг интегрирования по x

double h\_y = 0.001; // шаг интегрирования по y

stime = clock();

integral(a\_x, b\_x, a\_y, b\_y, h\_x, h\_y, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time;

double max\_time;

double avg\_time;

int numbExp = 10;

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << "execution time : " << avg\_time / numbExp << "; " <<

min\_time << "; " << max\_time << endl;

cout << "integral value : " << res << endl;

return 0;

}