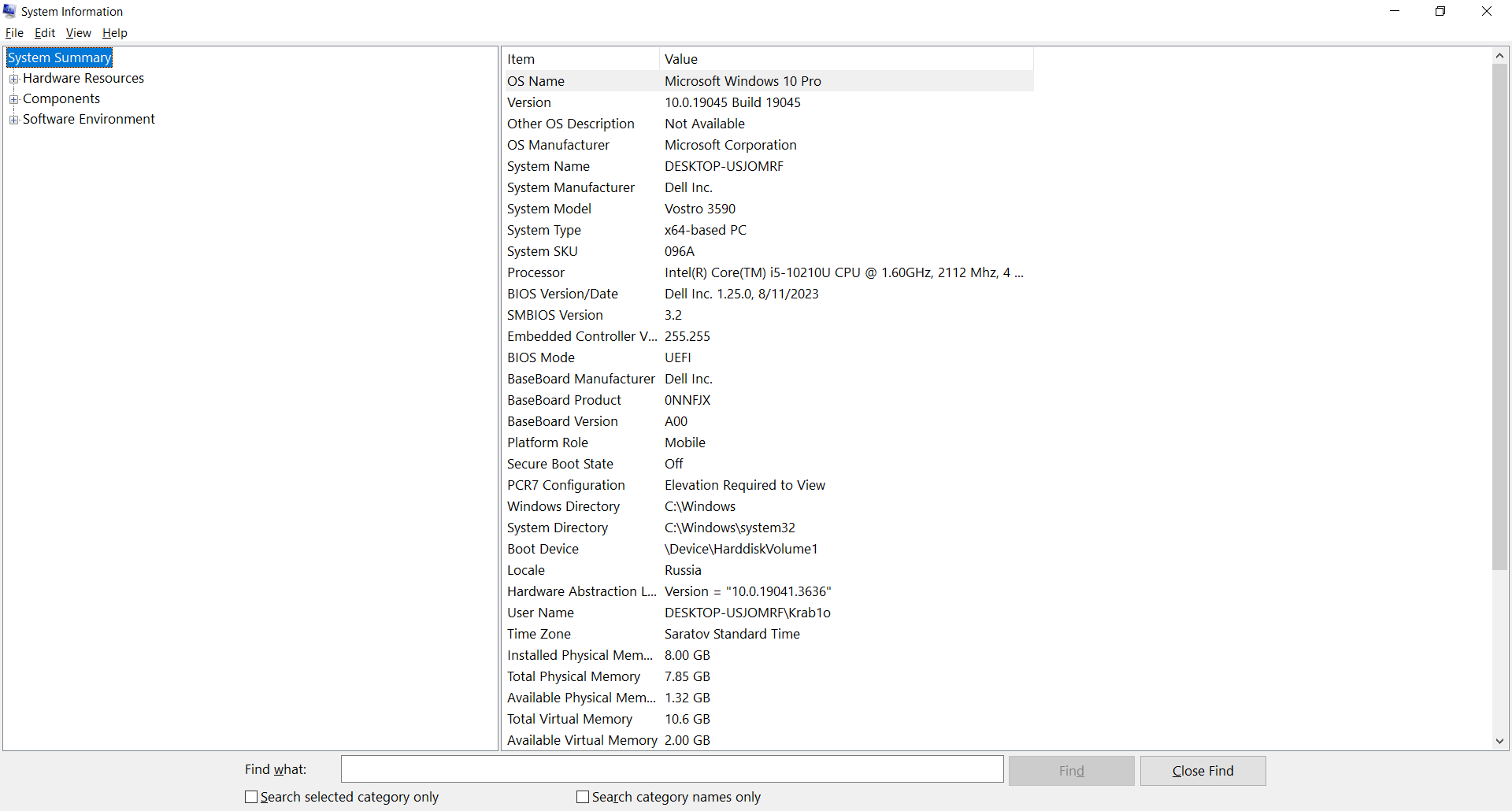
Задание выполнялось на ноутбуке со следующими параметрами:

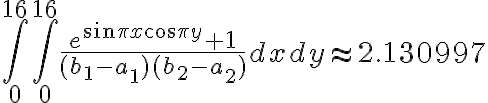


Из важного:

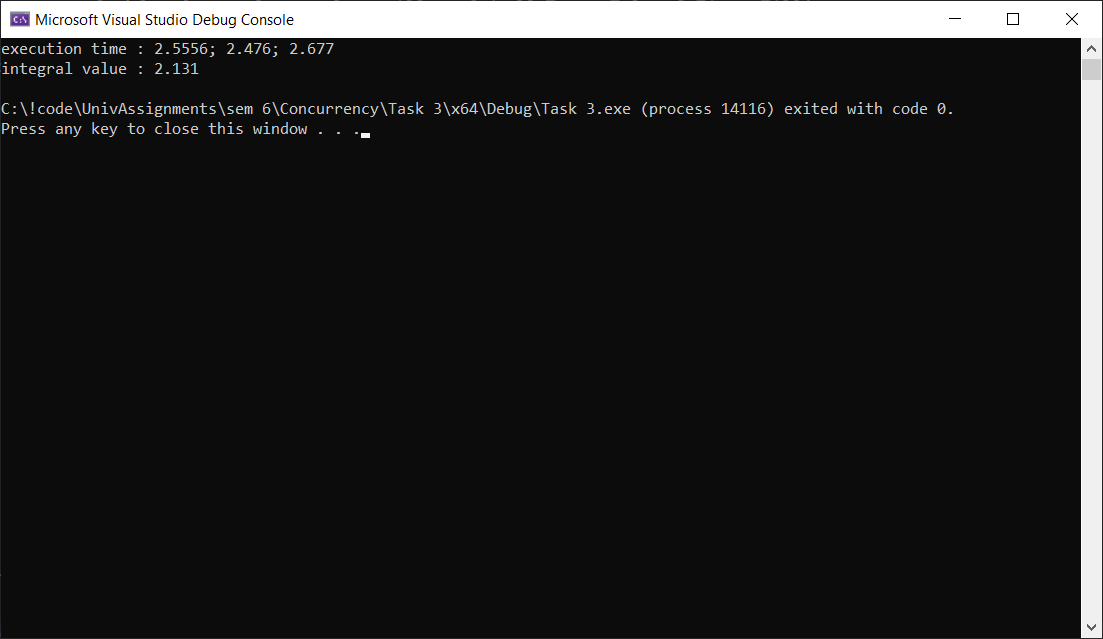
CPU: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz, 2112 Mhz, 4 Core(s), 8 Logical Processor(s)

RAM: 8.00 GB

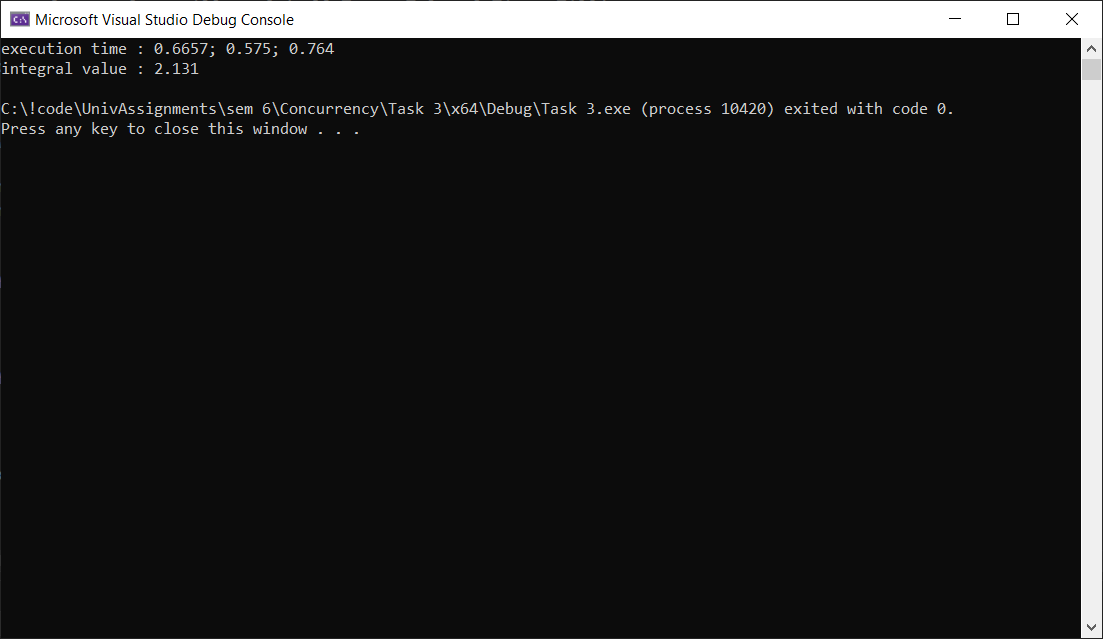
Задание. Реализовать подсчёт следующего интеграла (методом ячеек):



Сначала посчитаем интеграл без применения распараллеливания:



Самое эффективное распараллеливание OpenMP:





Аналогично с предыдущим заданием, MPI несколько проигрывает OpenMP из-за своих больших накладных расходов на распараллеливание процессов.

#include <iostream>

#include <mpi.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)

{

int i, j, n, m;

double sum1, sum2; // локальная переменная для подсчета интеграла

double x, y; // координата точки сетки

n = (int)((b\_x - a\_x) / h\_x); // количество точек сетки интегрирования

m = (int)((b\_y - a\_y) / h\_y); // количество точек сетки интегрирования

sum1 = 0.0;

sum2 = 0.0;

int NProc, ProcId;

MPI\_Comm\_size(MPI\_COMM\_WORLD, &NProc);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcId);

long long n\_x = n / NProc;

long long n\_y = (ProcId + 1) \* n\_x;

if (NProc == ProcId + 1) {

n\_y = n;

}

long long st = ProcId \* n\_x;

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

{

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

sum2 = 0.0;

for (j = 0; j < m; j++)

{

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

sum2 += f(x, y, a\_x, b\_x, a\_y, b\_y) \* h\_x \* h\_y;

}

sum1 += sum2;

}

MPI\_Reduce(&sum1, res, 1, MPI\_LONG\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

}

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 = 0.005; // 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()

{

MPI\_Init(NULL, NULL);

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;

}

int ProcId;

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcId);

if (ProcId == 0) {

// вывод результатов эксперимента

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

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

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

}

MPI\_Finalize();

return 0;

}