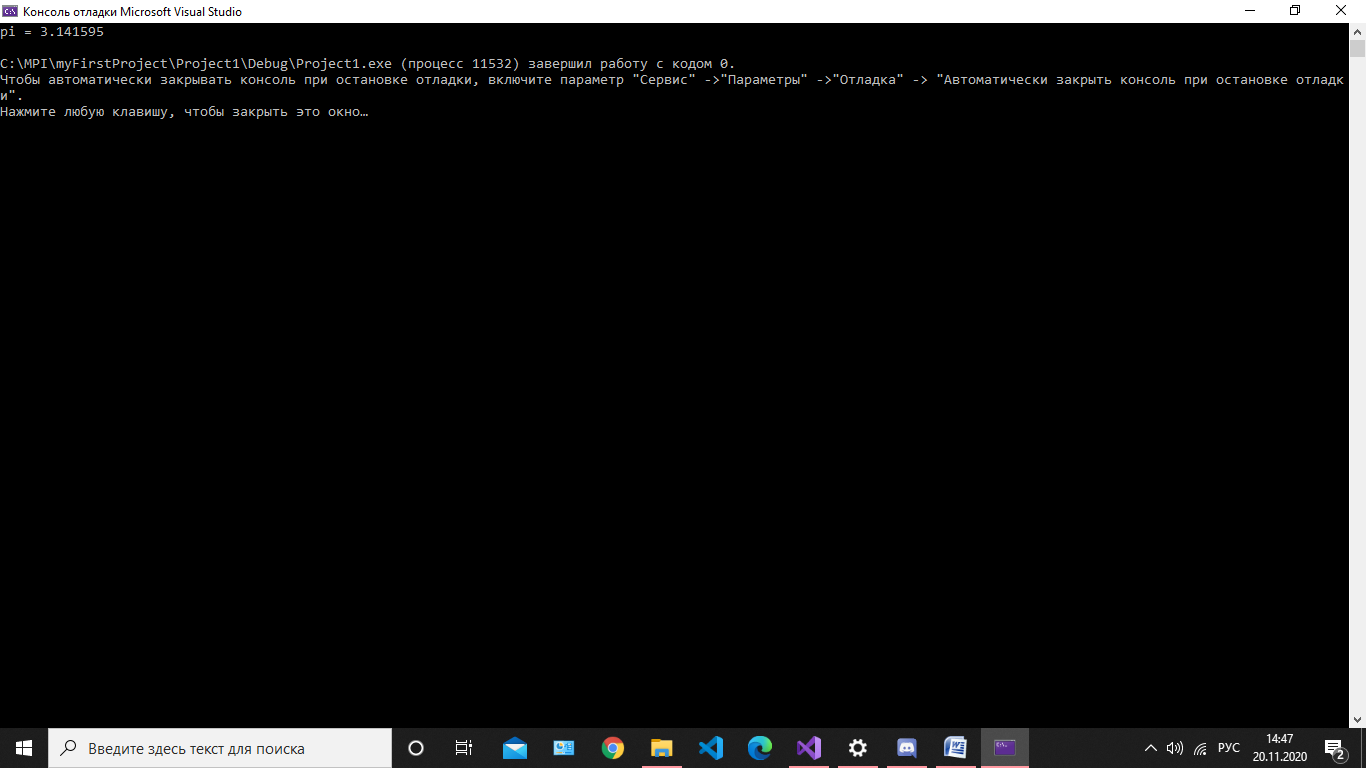
Иманбек Рустем информатика 18-5А

Задание 8. Распараллеливание циклов в OpenMP: программа «Число 𝜋» 1. Напишите OpenMP-программу, которая вычисляет число 𝜋 с точностью до N знаков после запятой.

|  |
| --- |
| #include <stdio.h>  double f(double y) { return(4.0 / (1.0 + y \* y)); }  int main()  {  double w, x, sum, pi;  int i;  int n = 1000000;  w = 1.0 / n;  sum = 0.0;  #pragma omp parallel for private(x) shared(w)\  reduction(+:sum)  for (i = 0; i < n; i++)  {  x = w \* (i - 0.5);  sum = sum + f(x);  }  pi = w \* sum;  printf("pi = %f\n", pi);  } |



Задание 9. Распараллеливание циклов в OpenMP: программа «Матрица» Напишите OpenMP-программу, которая вычисляет произведение двух квадратных матриц 𝐴 × 𝐵 = С размера 𝑛 × n

|  |
| --- |
| #include <omp.h>  #include <stdio.h>  #include <iostream>  #include <iomanip>  #include <time.h>  void matrixmult(int\*\* m1, int m1\_row, int m1\_col, int\*\* m2, int m2\_row, int m2\_col, int\*\* m3, int m3\_row, int m3\_col);  void matrixfill(int\*\* m, int row, int col)  {  for (int i = 0; i < row; ++i)  for (int j = 0; j < col; ++j)  {  std::cout << "Enter [ " << i << " ][ " << j << " ] element: ";  std::cin >> m[i][j];  }  }  void print\_matr(int\*\* m, int row, int col)  {  for (int i = 0; i < row; ++i)  {  for (int j = 0; j < col; ++j)  std::cout << std::setw(3) << m[i][j] << ' ';  std::cout << std::endl;  }  }  int main()  {  int row1 = 0;  int col1 = 0;  clock\_t start, stop;  std::cout << "Enter number of rows of the first matrix: ";  std::cin >> row1;  std::cout << "Enter number of columns of the first matrix: ";  std::cin >> col1;  int\*\* arr1 = new int\* [row1];  for (int i = 0; i < row1; ++i)  arr1[i] = new int[col1];  matrixfill(arr1, row1, col1);  int row2 = 0;  int col2 = 0;  std::cout << "Enter number of rows of the second matrix: ";  std::cin >> row2;  std::cout << "Enter number of columns of the second matrix: ";  std::cin >> col2;  int\*\* arr2 = new int\* [row2];  for (int i = 0; i < row2; ++i)  arr2[i] = new int[col2];  matrixfill(arr2, row2, col2);  int row3 = row1;  int col3 = col2;  int\*\* arr3 = new int\* [row3];  for (int i = 0; i < row3; ++i)  arr3[i] = new int[col3];  for (int i = 0; i < row3; ++i)  for (int j = 0; j < col3; ++j)  arr3[i][j] = 0;  std::cout << "\nFirst matrix: \n";  print\_matr(arr1, row1, col1);  std::cout << "\nSecond matrix: \n";  print\_matr(arr2, row2, col2);  start = clock();  //если число столбцов первой матрицы равно числу строк второй матрицы  if (col1 == row2)  matrixmult(arr1, row1, col1, arr2, row2, col2, arr3, row3, col3);  else  {  std::cerr << " Error! " << std::endl;  return 1;  }  std::cout << "\nResult matrix:" << std::endl;  print\_matr(arr3, row3, col3);  for (int i = 0; i < row1; i++)  delete[] arr1[i];  delete[] arr1;  for (int i = 0; i < row2; i++)  delete[] arr2[i];  delete[] arr2;  for (int i = 0; i < row3; i++)  delete[] arr3[i];  delete[] arr3;  stop = clock();  printf("Time=%f sec.\n", ((double)(stop - start) / 1000.0));  return 0;  }  void matrixmult(int\*\* m1, int m1\_row, int m1\_col, int\*\* m2, int m2\_row, int m2\_col, int\*\* m3, int m3\_row, int m3\_col)  {  omp\_set\_num\_threads(12);  int temp = 0;  int i = 0, j = 0, k = 0;  #pragma omp parallel  {  int num = omp\_get\_thread\_num();  #pragma omp parallel for schedule(static) private(k)  for (k = 0; k < m2\_col; ++k)  #pragma omp parallel for schedule(static) private(i)  for (i = 0; i < m1\_row; ++i)  #pragma omp parallel for schedule(static) shared(m1, m2, m3) private(j) reduction(+:temp)  for (j = 0; j < m1\_col; ++j)  {  temp += m1[i][j] \* m2[j][k];  m3[i][k] = temp;  }  }  } |

