Технологии параллельного программирования

Модуль 1. Введение. Технология OpenMP

Семинар 1

**Задание 1. Установка компилятора g++ и для него реализацию технологии OpenMP (как вариант, можно установить MS Visual Studio Community** [**https://visualstudio.microsoft.com/ru/free-developer-offers/**](https://visualstudio.microsoft.com/ru/free-developer-offers/) **)**

Скачиваем <https://www.msys2.org>

Устанавливаем g++:

Командные строки:

$ pacman -S mingw-w64-x86\_64-gcc

$ pacman -S mingw-w64-x86\_64-gdb

Устанавливаем OpenMP:

Скачиваем

<https://packages.msys2.org/package/mingw-w64-x86_64-openmp?repo=mingw64>

Командная строка:

$ pacman -S mingw-w64-x86\_64-openmp

**Задание 2. Пример программы умножения по разным циклам (без OpenMP)**

Приведенный программный код умножения матриц записать в файл \*.c, например, Matr.c

#include <iostream>

#include <time.h>

#include <stdlib.h>

using namespace std;

// Функция выделения памяти под 2-ный массив

double\*\* malloc\_array(long int n)

{

double\*\* matrix = new double\* [n];

for (int i = 0; i < n; i++)

matrix[i] = new double[n];

return matrix;

}

// Функция освобождения памяти

void free\_array(double\*\* matrix, long int n)

{

for (int i = 0; i < n; i++)

delete[] matrix[i];

delete[] matrix;

}

// Функция инициализации матрицы случайными числами из [0,1]

void rand\_init\_matrix(double\*\* matrix, long int n)

{

srand(time(NULL));

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

matrix[i][j] = rand() / RAND\_MAX;

}

// Функция обнуления матрицы

void zero\_init\_matrix(double\*\* matrix, long int n)

{

srand(time(NULL));

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

matrix[i][j] = 0.0;

}

int main()

{

const long int N = 1000;

double\*\* A, \*\* B, \*\* C;

// Выделение памяти под матрицы A,B,C

A = malloc\_array(N);

B = malloc\_array(N);

C = malloc\_array(N);

// Инициализация матриц

rand\_init\_matrix(A, N);

rand\_init\_matrix(B, N);

zero\_init\_matrix(C, N);

clock\_t t;

// Перемножение матриц с порядком циклов jki

t = clock();

for (int j = 0; j < N; j++)

for (int k = 0; k < N; k++)

for (int i = 0; i < N; i++)

C[i][j] += A[i][k] \* B[k][j];

t = clock() - t;

cout << "Time jki loops is " << t / CLOCKS\_PER\_SEC << " seconds" << endl;

// Перемножение матриц с порядком циклов ikj

zero\_init\_matrix(C, N);

t = clock();

for (int i = 0; i < N; i++)

for (int k = 0; k < N; k++)

for (int j = 0; j < N; j++)

C[i][j] += A[i][k] \* B[k][j];

t = clock() - t;

cout << "Time ikj loops is " << t / CLOCKS\_PER\_SEC << " seconds" << endl;

// Освобождение памяти занимаемой матрицами A,B,C

free\_array(A, N);

free\_array(B, N);

free\_array(C, N);

system("pause");

return 0;

}

После этого откомпилировать имеющимся компилятором (например, установленным g++) и запустить.

Например, следующие команды нужно выполнить:

- открыть в меню Пуск MSYS2 MINGW64

- сменить текущий каталог на каталог, в котором Вы сохранили исходный файл С с кодом:

$ cd /c/Users/Cristin/Desktop/ТПП/Матрицы

- откомпилировать код:

$ g++ Matr.c -fopenmp -o Matr.exe

- запустить полученный файл:

$ ./Matr.exe

Пояснить:

- структуру программы (кратко);

- причину разницы во времени умножения матриц по разным циклам.