***Этап 4)* разработка программы**

//-----ПРО. Лаб 4.

//-----Матвийчук Богдан ИВ-91

//-----a = max(B\*l + C\*(MO\*MK))

//-----18.04.2012

#include "stdafx.h"

#include <iostream>

#include <conio.h>

#include <windows.h>

#include <stdio.h>

#include <omp.h>

using std::cout;

using std::endl;

const int n = 8;

int p = 4;

int H = n / p;

int a=MININT;

int a1;

int a2;

int a3;

int a4;

int l;

int B[n], C[n];

int MO[n][n], MK[n][n];

int X[n];

int M[n];

int MXE[n][n];

int MXED[n][n];

// \* \* \* \* \* Ввод вектора

void VectorInput(int V[n])

{

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

{

V[i] = 1;

}

}

// \* \* \* \* \* Ввод матрицы

void MatrixInput(int M[n][n])

{

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

{

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

{

M[i][j] = 1;

}

}

}

// \* \* \* \* \* Вывод вектора

void VectorOutput(int V[n])

{

if (n <= 10)

{

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

{

cout<<V[i]<<" ";

}

cout<<endl;

}

}

// \* \* \* \* \* Вывод матрицы

void MatrixOutput(int M[n][n])

{

if (n < 10)

{

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

{

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

{

cout<<M[i][j]<<" ";

}

cout<<endl;

}

cout<<endl;

}

}

// \* \* \* \* \* Копирование вектора

void VectorCopy(int vector1[n], int vector2[n])

{

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

vector2[i] = vector1[i];

}

// \* \* \* \* \* Копирование матрицы

void MatrixCopy(int matrix1[n][n], int matrix2[n][n])

{

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

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

matrix2[i][j] = matrix1[i][j];

}

// \* \* \* \* \* Умножение матриц

void MatrixMnozh(int one, int two, int matrOP[n][n], int matr[n][n], int matrR[n][n])

{

for (int i = one; i < two; i++)

{

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

{

matrR[i][j] = 0;

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

{

matrR[i][j] += matrOP[k][j] \* matr[i][k];

}

}

}

}

// \* \* \* \* \* Умножение матрицы на вектор

void MatrixVecMnozh(int one, int two, int vecOP[n], int matr[n][n], int vecR[n])

{

for (int i = one; i < two; i++)

{

vecR[i] = 0;

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

{

vecR[i] += vecOP[j] \* matr[i][j];

}

}

}

// \* \* \* \* \* Сложение векторов

void VecSlozh(int one, int two, int vec1[n], int vec2[n], int vecR[n])

{

for (int i = one; i < two; i++)

{

vecR[i] = vec1[i] + vec2[i];

}

}

void VecSkal(int one, int two, int vec[n], int b, int vecR[n])

{

for (int i = one; i < two; i++)

{

vecR[i] = vec[i]\*b;

}

}

// \* \* \* \* \* Сортировка вектора

void MaxVec(int one, int two, int vec[n],int ab)

{

for (int i = one; i < two; i++)

{

for (int j = one; j < two - 1; j++)

{

if (vec[j] < vec[j + 1])

{

ab=vec[j];

}

}

}

}

// \* \* \* \* \* Сортировка вектора слиянием

void MaxVectorSln(int one, int two, int ab)

{

if (a < ab) a = ab;

}

// \* \* \* \* \* \* \* \* \* \* TASK T1 \* \* \* \* \* \* \* \* \* \*

void T1\_B1()

{

cout<<"T1 started"<<endl;

VectorInput(B);

VectorInput(C);

l=1;

}

void T1\_B2()

{

int C1[n];

int MK1[n][n];

#pragma omp critical

{

memmove(C1, C, sizeof(C1));

memmove(MK1, MK, sizeof(MK1));

}

MatrixMnozh(0, H, MK1, MO, MXE);

MatrixVecMnozh(0, H, C1, MXE, X);

VecSkal (0,H,B,l,M);

VecSlozh(0, H, M, X, M);

}

void T1\_B3()

{

MaxVec(0, H, M,a1);

}

void T1\_B4()

{

MaxVectorSln(0, H, a1);

}

void T1\_B5()

{

cout<<a<<" ";

cout<<"T1 finished."<<endl;

cout<<endl;

}

// \* \* \* \* \* \* \* \* \* \* TASK T2 \* \* \* \* \* \* \* \* \* \*

void T2\_B1()

{

cout<<"T2 started"<<endl;

MatrixInput(MO);

}

void T2\_B2()

{

int C2[n];

int MK2[n][n];

#pragma omp critical

{

memmove(C2, C, sizeof(C2));

memmove(MK2, MK, sizeof(MK2));

}

MatrixMnozh(0, H, MK2, MO, MXE);

MatrixVecMnozh(0, H, C2, MXE, X);

VecSkal (0,H,B,l,M);

VecSlozh(0, H, M, X, M);

}

void T2\_B3()

{

MaxVec(0, H, M,a2);

}

void T2\_B4()

{

MaxVectorSln(0, H, a2);

cout<<"T2 finished."<<endl;

cout<<endl;

}

// \* \* \* \* \* \* \* \* \* \* TASK T3 \* \* \* \* \* \* \* \* \* \*

void T3\_B1()

{

cout<<"T3 started"<<endl;

}

void T3\_B2()

{

int C3[n];

int MK3[n][n];

#pragma omp critical

{

memmove(C3, C, sizeof(C3));

memmove(MK3, MK, sizeof(MK3));

}

MatrixMnozh(0, H, MK3, MO, MXE);

MatrixVecMnozh(0, H, C3, MXE, X);

VecSkal (0,H,B,l,M);

VecSlozh(0, H, M, X, M);

}

void T3\_B3()

{

MaxVec(0, H, M,a3);

}

void T3\_B4()

{

MaxVectorSln(0, H, a3);

cout<<"T3 finished."<<endl;

cout<<endl;

}

// \* \* \* \* \* \* \* \* \* \* TASK T4 \* \* \* \* \* \* \* \* \* \*

void T4\_B1()

{

cout<<"T4 started"<<endl;

MatrixInput(MK);

}

void T4\_B2()

{

int C4[n];

int MK4[n][n];

#pragma omp critical

{

memmove(C4, C, sizeof(C4));

memmove(MK4, MK, sizeof(MK4));

}

MatrixMnozh(0, H, MK4, MO, MXE);

MatrixVecMnozh(0, H, C4, MXE, X);

VecSkal (0,H,B,l,M);

VecSlozh(0, H, M, X, M);

}

void T4\_B3()

{

MaxVec(0, H, M,a4);

}

void T4\_B4()

{

MaxVectorSln(0, H, a4);

cout<<"T4 finished."<<endl;

cout<<endl;

}

// \* \* \* \* \* \* \* \* \* \* MAIN \* \* \* \* \* \* \* \* \* \*

int \_tmain(int argc, \_TCHAR\* argv[])

{

cout<<"\* - Laba4 started"<<endl;

omp\_set\_num\_threads(p);

#pragma comment(linker, "/STACK:200000000")

#pragma omp parallel

{

switch (omp\_get\_thread\_num())

{

case 0: T1\_B1();

break;

case 1: T2\_B1();

break;

case 2: T3\_B1();

break;

case 3: T4\_B1();

break;

}

#pragma omp barrier

switch (omp\_get\_thread\_num())

{

case 0: T1\_B2();

break;

case 1: T2\_B2();

break;

case 2: T3\_B2();

break;

case 3: T4\_B2();

break;

}

#pragma omp barrier

switch (omp\_get\_thread\_num())

{

case 0: T1\_B3();

break;

case 1: T2\_B3();

break;

case 2: T3\_B3();

break;

case 3: T4\_B3();

break;

}

#pragma omp barrier

switch (omp\_get\_thread\_num())

{

case 0: T1\_B4();

break;

case 2: T3\_B4();

break;

}

#pragma omp barrier

if (omp\_get\_thread\_num() == 3)

T1\_B5();

}

cout<<"\* - Laba4 finished."<<endl;

getchar();

}