1. **Лістинг програми ПРГ1.**

// course work.part 2

// A=sort(B+C+a\*Z\*(MO\*MK))

// Romas Andriy IO-01

// 15.04.2013

#include <iostream>

#include <windows.h>

#include <time.h>

using namespace std;

CRITICAL\_SECTION cs;

HANDLE mutexx;

HANDLE event1, event2, event3;

HANDLE event11, event21, event31, event41, event51, event61;

HANDLE sem1, sem2, sem3, sem5, sem6;

const int n = 900;

const int p = 6;

int h = n / p;

int value = 1;

int MX[n][n];

int E[n];

int D[n];

int V[n];

int MO[n][n];

int MK[n][n];

int B[n];

int C[n];

int Z[n];

int A[n];

int a;

DWORD f1(void) {

std::cout << "Thread1 started" << std::endl;

// input a,B,C,Z

a = value;

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

B[i] = value;

C[i] = value;

Z[i] = value;

}

// signal notifies end of input

SetEvent(event1);

// wait for other threads end input

WaitForSingleObject(event2, INFINITE);

WaitForSingleObject(event3, INFINITE);

// use mutex for local copy

int MO1[n][n];

int Z1[n];

EnterCriticalSection(&cs);

int a1 = a;

LeaveCriticalSection(&cs);

WaitForSingleObject(mutexx, INFINITE);

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

Z1[i] = Z[i];

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

MO1[i][j] = MO[i][j];

}

ReleaseMutex(mutexx);

// calculating

int Z11[n];

for (int j = 0; j < h; j++) {

Z11[j] = 0;

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

MX[i][j] = 0;

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

MX[i][j] += MO1[i][k] \* MK[k][j];

Z11[j] += a1\*Z1[j]\*MX[r][j];

}

V[j] += Z11[j] + B[j] + C[j];

}

for (int i = 0; i < h - 1; i++)

for (int j = i + 1; j < h; j++)

if (V[i] > V[j]) {

int temp = V[j];

V[j] = V[i];

V[i] = temp;

}

//----------------------------------

ReleaseSemaphore(sem1, 1, NULL);

cout << "Thread1 finished" << endl;

return 0;

}

DWORD f2(void) {

std::cout << "Thread2 started" << std::endl;

// wait for other threads end input

WaitForSingleObject(event2, INFINITE);

WaitForSingleObject(event1, INFINITE);

WaitForSingleObject(event3, INFINITE);

int MO2[n][n];

int Z2[n];

EnterCriticalSection(&cs);

int a2 = a;

LeaveCriticalSection(&cs);

WaitForSingleObject(mutexx, INFINITE);

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

Z2[i] = Z[i];

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

MO2[i][j] = MO[i][j];

}

boolean flg1 = ReleaseMutex(mutexx);

int Z22[n];

for (int j = h; j < 2 \* h; j++) {

Z22[j] = 0;

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

MX[i][j] = 0;

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

MX[i][j] += MO2[i][k] \* MK[k][j];

Z22[j] += a2\*Z2[j]\*MX[r][j];

}

V[j] += Z22[j] + B[j] + C[j];

}

for (int i = h; i < 2 \* h - 1; i++)

for (int j = i + 1; j < 2 \* h; j++)

if (V[i] > V[j]) {

int temp = V[j];

V[j] = V[i];

V[i] = temp;

}

flg1 = ReleaseSemaphore(sem2, 1, NULL);

std::cout << "Thread2 finished" << std::endl;

return 0;

}

DWORD f3(void) {

std::cout << "Thread3 started" << std::endl;

// wait for other threads end input

WaitForSingleObject(event1, INFINITE);

WaitForSingleObject(event2, INFINITE);

WaitForSingleObject(event3, INFINITE);

int MO3[n][n];

int Z3[n];

EnterCriticalSection(&cs);

int a3 = a;

LeaveCriticalSection(&cs);

WaitForSingleObject(mutexx, INFINITE);

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

Z3[i] = Z[i];

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

MO3[i][j] = MO[i][j];

}

ReleaseMutex(mutexx);

int Z33[n][n];

for (int j = 2 \* h; j < 3 \* h; j++) {

Z33[j] = 0;

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

MX[i][j] = 0;

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

MX[i][j] += MO3[i][k] \* MK[k][j];

Z33[j] += a3\*Z3[j]\*MX[r][j];

}

V[j] += Z33[j] + B[j] + C[j];

}

for (int i = 2 \* h; i < 3 \* h - 1; i++)

for (int j = i + 1; j < 3 \* h; j++)

if (V[i] > V[j]) {

int temp = V[j];

V[j] = V[i];

V[i] = temp;

}

ReleaseSemaphore(sem3, 1, NULL);

std::cout << "Thread3 finished" << std::endl;

return 0;

}

DWORD f4(void) {

std::cout << "Thread4 started" << std::endl;

// wait for other threads end input

WaitForSingleObject(event1, INFINITE);

WaitForSingleObject(event2, INFINITE);

WaitForSingleObject(event3, INFINITE);

int MO4[n][n];

int Z4[n];

EnterCriticalSection(&cs);

int a4 = a;

LeaveCriticalSection(&cs);

WaitForSingleObject(mutexx, INFINITE);

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

Z4[i] = 0;

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

MO4[i][j] = MO[i][j];

}

ReleaseMutex(mutexx);

int Z44[n];

for (int j = 3 \* h; j < 4 \* h; j++) {

Z44[j] = 0;

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

MX[i][j] = 0;

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

MX[i][j] += MO4[i][k] \* MK[k][j];

Z44[j] += a4\*Z4\*MX[r][j];

}

V[j] += Z44 + B[j] + C[j];

}

for (int i = 3 \* h; i < 4 \* h - 1; i++) {

for (int j = i + 1; j < 4 \* h; j++) {

if (V[i] > V[j]) {

int temp = V[j];

V[j] = V[i];

V[i] = temp;

}

}

}

WaitForSingleObject(sem1, INFINITE);

WaitForSingleObject(sem2, INFINITE);

WaitForSingleObject(sem3, INFINITE);

WaitForSingleObject(sem5, INFINITE);

WaitForSingleObject(sem6, INFINITE);

int index[p];

index[0] = 0;

index[1] = h;

index[2] = 2 \* h;

index[3] = 3 \* h;

index[4] = 4 \* h;

index[5] = 5 \* h;

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

int min = V[index[0]];

int minIndex = 0;

for (int j = 1; j < p; j++) {

if (V[index[j]] < min) {

min = V[index[j]];

minIndex = j;

}

}

A[i] = V[minIndex];

index[minIndex]++;

}

if (n <= 8) {

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

cout << A[i] << endl;

}

std::cout << "Thread4 finished" << std::endl;

return 0;

}

DWORD f5(void) {

std::cout << "Thread5 started" << std::endl;

// input MO

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

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

MO[i][j] = value;

// signal notifies end of input

SetEvent(event2);

// wait for other threads end input

WaitForSingleObject(event3, INFINITE);

WaitForSingleObject(event1, INFINITE);

int MO5[n][n];

int Z5[n];

EnterCriticalSection(&cs);

int a5 = a;

LeaveCriticalSection(&cs);

WaitForSingleObject(mutexx, INFINITE);

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

Z5[i] = Z[i];

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

MO5[i][j] = MO[i][j];

}

ReleaseMutex(mutexx);

int Z55[n];

for (int j = 4 \* h; j < 5 \* h; j++) {

Z55[j] = 0;

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

MX[i][j] = 0;

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

MX[i][j] += MO5[i][k] \* MK[k][j];

Z55[j] += a5\*Z5\*MX[r][j];

}

V[j] += Z55 + B[j] + C[j];

}

for (int i = 4 \* h; i < 5 \* h - 1; i++)

for (int j = i + 1; j < 5 \* h; j++)

if (V[i] > V[j]) {

int temp = V[j];

V[j] = V[i];

V[i] = temp;

}

ReleaseSemaphore(sem5, 1, NULL);

std::cout << "Thread5 finished" << std::endl;

return 0;

}

DWORD f6(void) {

std::cout << "Thread6 started" << std::endl;

// input MT

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

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

MK[i][j] = value;

// signal notifies end of input

SetEvent(event3);

// wait for other threads end input

WaitForSingleObject(event2, INFINITE);

WaitForSingleObject(event1, INFINITE);

int MO6[n][n];

int Z6[n];

EnterCriticalSection(&cs);

int a6 = a;

LeaveCriticalSection(&cs);

WaitForSingleObject(mutexx, INFINITE);

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

Z6[i] = 0;

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

MO6[i][j] = MO[i][j];

}

ReleaseMutex(mutexx);

int Z66[n];

for (int j = 5 \* h; j < n; j++) {

Z66[j] = 0;

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

MX[i][j] = 0;

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

MX[i][j] += MO6[i][k] \* MK[k][j];

Z66 += a6\*Z6[j]\*MX[r][j];

}

V[j] += Z66 + B[j] + C[j];

}

for (int i = 5 \* h; i < n - 1; i++) {

for (int j = i + 1; j < n; j++) {

if (V[i] > V[j]) {

int temp = V[j];

V[j] = V[i];

V[i] = temp;

}

}

}

ReleaseSemaphore(sem6, 1, NULL);

std::cout << "Thread6 finished" << std::endl;

return 0;

}

int main() {

cout << "main started" << endl;

HANDLE PT1, PT2, PT3, PT4, PT5, PT6;

DWORD tid;

DWORD stacksize = 99999999;

event1 = CreateEvent(NULL, 1, 0, NULL);

event2 = CreateEvent(NULL, 1, 0, NULL);

event3 = CreateEvent(NULL, 1, 0, NULL);

event11 = CreateEvent(NULL, 1, 0, NULL);

event21 = CreateEvent(NULL, 1, 0, NULL);

event31 = CreateEvent(NULL, 1, 0, NULL);

event41 = CreateEvent(NULL, 1, 0, NULL);

event51 = CreateEvent(NULL, 1, 0, NULL);

event61 = CreateEvent(NULL, 1, 0, NULL);

mutexx = CreateMutex(NULL, 0, NULL);

sem1 = CreateSemaphore(NULL, 0, 1, NULL);

sem2 = CreateSemaphore(NULL, 0, 1, NULL);

sem3 = CreateSemaphore(NULL, 0, 1, NULL);

sem5 = CreateSemaphore(NULL, 0, 1, NULL);

sem6 = CreateSemaphore(NULL, 0, 1, NULL);

InitializeCriticalSection(&cs);

clock\_t t1, t2;

string t;

cout << "Press enter to start" << endl;

getline(cin, t, '\n');

t1 = clock();

PT1 = CreateThread(NULL, stacksize, (LPTHREAD\_START\_ROUTINE) f1, NULL, 0,

&tid);

PT2 = CreateThread(NULL, stacksize, (LPTHREAD\_START\_ROUTINE) f2, NULL, 0,

&tid);

PT3 = CreateThread(NULL, stacksize, (LPTHREAD\_START\_ROUTINE) f3, NULL, 0,

&tid);

PT4 = CreateThread(NULL, stacksize, (LPTHREAD\_START\_ROUTINE) f4, NULL, 0,

&tid);

PT5 = CreateThread(NULL, stacksize, (LPTHREAD\_START\_ROUTINE) f5, NULL, 0,

&tid);

PT6 = CreateThread(NULL, stacksize, (LPTHREAD\_START\_ROUTINE) f6, NULL, 0,

&tid);

WaitForSingleObject(PT1, INFINITE);

WaitForSingleObject(PT2, INFINITE);

WaitForSingleObject(PT3, INFINITE);

WaitForSingleObject(PT4, INFINITE);

WaitForSingleObject(PT5, INFINITE);

WaitForSingleObject(PT6, INFINITE);

t2 = clock();

cout << "main finished" << endl;

cout << t2 - t1 << endl;

return 0;

}