Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра Систем Штучного Інтелекту

Звіт

до лабораторної роботи № 3

з дисципліни

Операційні системи

на тему:

“Виконання задачі в декількох потоках в ОС Windows”

Виконав: студент КН-217

Ратушняк Денис

Прийняв: доцент каф. СШІ

Кривенчук Ю. П.

Львів – 2022

Мета роботи: Навчитись реалізовувати розпаралелювання алгоритмів за допомогою багатопоточності в ОС Windows з використанням функцій WinAPI.

Завдання:

26. [Складність 4] Знайти**, на якій діагоналі матриці середнє значення елементів є максимальним.** Діагоналлю вважати будь-яку діагональну лінію, що проходить крізь матрицю.

Текст програми-генератора тестових даних(C++):

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
long long mod = 1e9+7;
ll n,m,k,max_number;
int main()
{
    ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);srand(clock());
    cin >> n >> m >> max_number;max_number++;
    vector< ll > random(max(n, m));
    for(int i = 0; i < random.size(); ++i) random[i] = i;
    vector< vector<ll> > a(n, vector<ll>(m,0));
    for(int i = 0 ; i < n; ++i){
        for(int j = 0; j < m; ++j){
            //random_shuffle(random.begin(),random.end());
            a[i][j] = (rand() * random[abs(i-j)] + rand() + rand() + rand()) % max_number;
            //a[i][j] = i + j + 5;
        }
    }
    string out_version = to_string(n) + "_" + to_string(m) + "_" + to_string(max_number-1) + "_out.txt";
    string in_version = to_string(n) + "_" + to_string(m) + "_" + to_string(max_number-1) + "_in.txt";
    string result_path = "A:\\T\\3 term\\Operating Systems\\3lab\\results_right\\" + out_version;
    string test_path = "A:\\T\\3 term\\Operating Systems\\3lab\\tests\\" + in_version;
    ofstream in(test_path);ofstream out(result_path);
    in << n << " " << m   << " " << max_number-1 << "\n\n";
    for(int i = 0; i < n; ++i){
        for(int j = 0; j < m; ++j)
            in << a[i][j] << " ";
        in << "\n";
    }
    in << "\n";
    ll diag_cnt = 2 * n + 2 * m - 2;
    vector<pair<ll,ll>> diag(diag_cnt);
    vector< pair< pair<ll,ll>, pair<ll,ll> > > points_for_diag(diag_cnt);
    ll x1 = 0;ll y1 = 0;ll x2 = 0;ll y2 = 0;
    for(int i = 0; i < diag_cnt/2; ++i){
        points_for_diag[i] = {{x1, y1}, {x2, y2}};
        if(x1 == n - 1) y1++;
        else x1++;

        if(y2 == m - 1) x2++;
        else y2++;
    }
    x1 = 0;y1 = m-1;x2 = 0;y2 = m-1;
    for(int i = diag_cnt/2; i < diag_cnt; ++i){
```

```
            points_for_diag[i] = {{x1, y1}, {x2, y2}};
            if(y1 == 0) x1++;
            else y1--;
            if(x2 == n - 1) y2--;
            else x2++;
        }
        //for(int i = 0; i < diag_cnt; ++i) cout << points_for_diag[i].first.first << " " << points_for_diag[i].first.second << " " << points_for_diag[i].second.first << " " << points_for_diag[i].second.second << "\n";
        ll add = n + 2*m - 2;
        for(int i = 0; i < n; ++i){
            for(int j = 0; j < m; ++j){
                diag[i+j].first += a[i][j];
                diag[i+j].second ++;
                diag[i-j+add].first += a[i][j];
                diag[i-j+add].second ++;
            }
        }
        ll ans = 0;
        for(int i = 1 ; i < diag.size(); ++i) if(diag[i].first * diag[ans].second > diag[i].second * diag[ans].first) ans = i;
        long double result = (long double)diag[ans].first / (long double)diag[ans].second;
        out << ans << " ";
        out << fixed << setprecision(5) << result << "\n";
        out << points_for_diag[ans].first.first+1 << " " << points_for_diag[ans].first.second+1 << " " << points_for_diag[ans].second.first+1 << " " << points_for_diag[ans].second.second+1 << "\n";
        return 0;
}
```

Текст програми-рішення задачі. (C++):

```
#include <bits/stdc++.h>
#include <windows.h>
#include <tchar.h>

using namespace std;
typedef long long ll;
typedef long double ld;
typedef vector< vector<ll> > matrix;
///HERE YOU CAN CHANGE INPUT
const int TOTAL_THREADS = 16;
string version = "3_3_10";
ll type_of_task = 1;
ll low_priority_thread = 3;
ll high_priority_thread = 7;
///HERE YOU CAN CHANGE INPUT
const long long mod = 1e9+7;

typedef struct MyData
{
    int startx, starty, step, type;
    int threadNum;
} MYDATA, *PMYDATA;

//PMYDATA pDataArray[TOTAL_THREADS];
MYDATA pDataArray[TOTAL_THREADS];
DWORD    dwThreadIdArray[TOTAL_THREADS];
HANDLE   hThreadArray[TOTAL_THREADS];
HANDLE stackTh[64];

matrix A;
ll n,m,k,max_number;
ll add;
```

```
vector < pair<ll,ll> > diag[TOTAL_THREADS];

vector< pair< pair<ll,ll>, pair<ll,ll> > > points_for_diag;

void solve(int sx, int sy, int step, int type, int num)
{
    if(type == 1)
    {
        int i = sx;
        int j = sy;
        while(step--){
            diag[num][i+j].first += A[i][j];
            diag[num][i+j].second ++;
            diag[num][i-j+add].first += A[i][j];
            diag[num][i-j+add].second ++;
            //cout << i << " " << j << endl;
            //cout << num << " " << step << endl;
            j++;
            if(j == m){
                j = 0;
                i ++;
                if(i == n) break;
            }
        }
    }
    else
    {
        int now = sx * n + sy;

        while(now < n * m){
            int i = now / m;
            int j = now % m;

            diag[num][i+j].first += A[i][j];
            diag[num][i+j].second ++;

            diag[num][i-j+add].first += A[i][j];
            diag[num][i-j+add].second ++;
            //cout << num << " " << now << " " << step << endl;
            now += step;
        }
    }

}
ld threadTime[TOTAL_THREADS];

DWORD WINAPI MyThreadFunction(LPVOID lpParam)
{
    LARGE_INTEGER st, en, fq;
    QueryPerformanceFrequency(&fq);
    QueryPerformanceCounter(&st);

    PMYDATA pDataVar;

    pDataVar = (PMYDATA)lpParam;
    MYDATA DataVar = *pDataVar;
    //cout << "HERE IS " << " " << pDataVar->threadNum << " thread" << endl;
    solve(DataVar.startx, DataVar.starty, DataVar.step, DataVar.type, DataVar.threadNum);
    QueryPerformanceCounter(&en);

    ld currtime = (en.QuadPart-st.QuadPart)*1000.0/fq.QuadPart;
    threadTime[pDataVar->threadNum] = currtime;
```

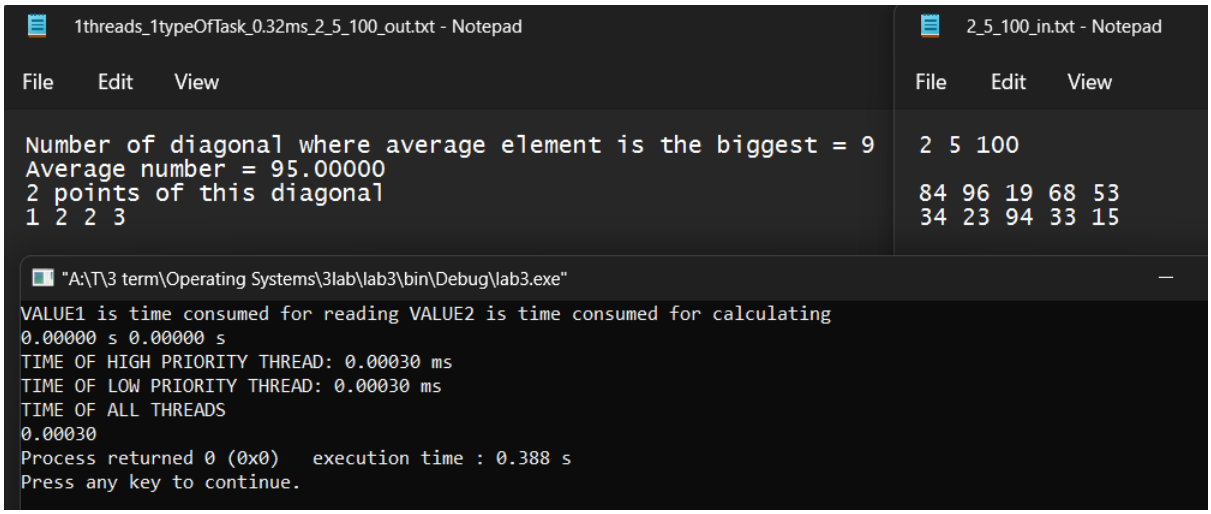
```
    return 0;
}

void print(matrix &a)
{
    for(int i = 0; i < a.size(); ++ i)
    {
        for(int j = 0; j < a[i].size(); ++ j)
        {
            cout << a[i][j] << " ";
        }
        cout << "\n";
    }
    cout << "\n";
}

void print(matrix &a, ofstream &output)
{
    for(int i = 0; i < a.size(); ++ i)
    {
        for(int j = 0; j < a[i].size(); ++ j)
        {
            output << a[i][j] << " ";
        }
        output << "\n";
    }
    output << "\n";
}

void read(matrix &a, ifstream &input)
{
    for(int i = 0; i < a.size(); ++ i)
    {
        for(int j = 0; j < a[i].size(); ++ j)
        {
            input >> a[i][j];
        }
    }
}


void calc_points()
{
    points_for_diag.resize(2 * m + 2 * n - 2);
    ll x1,y1,x2,y2;
    x1 = y1 = x2 = y2 = 0;

    for(int i = 0; i < (2 * m + 2 * n - 2)/2; ++i)
    {
        points_for_diag[i] = {{x1, y1}, {x2, y2}};
        if(x1 == n - 1) y1++;
        else x1++;

        if(y2 == m - 1) x2++;
        else y2++;
    }

    x1 = x2 = 0;
    y1 = y2 = m-1;

    for(int i = (2 * m + 2 * n - 2)/2; i < (2 * m + 2 * n - 2); ++i)
    {
```

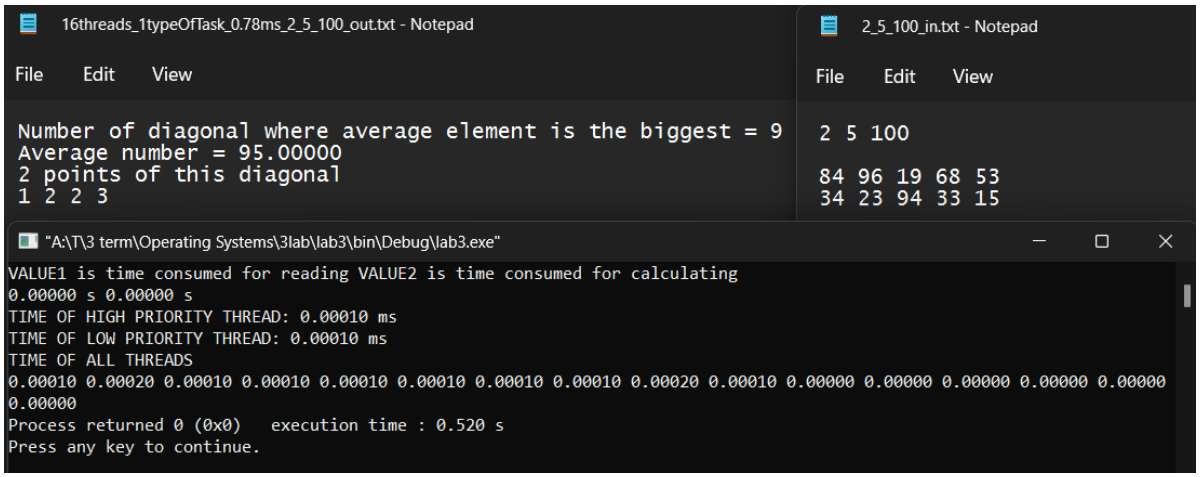
```
            points_for_diag[i] = {{x1, y1}, {x2, y2}};
            if(y1 == 0) x1++;
            else y1--;

            if(x2 == n - 1) y2--;
            else x2++;
        }
}

int main()
{
    string test_path = "A:\\T\\3 term\\Operating Systems\\3lab\\tests\\" + version + "_in.txt";

    low_priority_thread %= TOTAL_THREADS;
    high_priority_thread %= TOTAL_THREADS;
    if(low_priority_thread == high_priority_thread) low_priority_thread--;
    if(low_priority_thread == -1) low_priority_thread = 1;
    if(low_priority_thread == TOTAL_THREADS) low_priority_thread = 0;

    ifstream input(test_path);
    ld t0 = clock();
    input >> n >> m >> max_number;
    add = n + 2*m - 2;
    calc_points();
    A.resize(n);
    for(int i = 0; i < n; ++i) A[i].resize(m,0);

    read(A, input);
    //print(A);

    ll step1 = (n * m / TOTAL_THREADS);
    if(!step1) step1++;


    LARGE_INTEGER st, en, fq;
    QueryPerformanceFrequency(&fq);
    QueryPerformanceCounter(&st);

    ld t1 = clock();
    ll now1 = 0;
    ll now2 = 0;

    ll needThreads = min(ll(n * m), ll(TOTAL_THREADS));
    ll step2 = needThreads;
    for(int i = 0; i < needThreads; ++i)
    {
        diag[i].resize(2 * n + 2 * m - 2, {0, 0});
        if(i%64 == 0) WaitForMultipleObjects(64, stackTh, TRUE, INFINITE);

        //pDataArray[i] = (PMYDATA) HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
sizeof(MYDATA));

        //if(pDataArray[i] == NULL) ExitProcess(2);

        pDataArray[i].threadNum = i;
        pDataArray[i].type = type_of_task;

        if(type_of_task == 1)
        {
            pDataArray[i].startx = now1 / m;
            pDataArray[i].starty = now1 % m;
            pDataArray[i].step = step1;
```

```
            now1 += step1;
        }
        else
        {
            pDataArray[i].startx = now2 / m;
            pDataArray[i].starty = now2 % m;
            pDataArray[i].step = step2;
            now2 ++;
        }

        hThreadArray[i] = CreateThread(
                                NULL,                   // default security attributes
                                0,                      // use default stack size
                                MyThreadFunction,       // thread function name
                                &pDataArray[i],         // argument to thread function
                                0,                      // use default creation flags
                                &dwThreadIdArray[i]);   // returns the thread identifier

        stackTh[i%64] = hThreadArray[i];

       // if(hThreadArray[i] == NULL) ExitProcess(3);

        if(i == high_priority_thread)
            SetThreadPriority(hThreadArray[i],THREAD_PRIORITY_HIGHEST);


        if(i == low_priority_thread)
            SetThreadPriority(hThreadArray[i],THREAD_PRIORITY_LOWEST);

    }

    WaitForMultipleObjects(min((ll)64, needThreads), stackTh, TRUE, INFINITE);

    for(int i = 0; i < needThreads; i++)
    {
        CloseHandle(hThreadArray[i]);
    }
    for(int i = 1; i < needThreads; ++i)
    {
        for(int j = 0; j < diag[0].size(); ++j){
            diag[0][j].first += diag[i][j].first;
            diag[0][j].second += diag[i][j].second;
            //cout << i << " " << j << " " << diag[i][j].first << " " << diag[i][j].second << endl;
        }
    }
    ll ans = 0;
    for(int i = 1 ; i < diag[0].size(); ++i){
        if(diag[0][i].first * diag[0][ans].second > diag[0][i].second * diag[0][ans].first) ans = i;
    }
   // for(int i = 0; i < diag[0].size(); ++i) cout << diag[0][i].first << " " << diag[0][i].second << endl;
    ld result = (ld)diag[0][ans].first / (ld)diag[0][ans].second;
    QueryPerformanceCounter(&en);
    ld t2 = clock();

    ld process_time = (en.QuadPart-st.QuadPart)*1000.0/fq.QuadPart;
    long long inttime = round(process_time * 100);
    long long dectime = inttime % 100;
    inttime /= 100;
    string result_path = "A:\\T\\3 term\\Operating Systems\\3lab\\results_prob_right\\" +
to_string(TOTAL_THREADS);
    result_path += "threads_" + to_string(type_of_task) + "typeOfTask_" + to_string(inttime) + "." +
to_string(dectime) + "ms_" + version + "_out.txt";
```

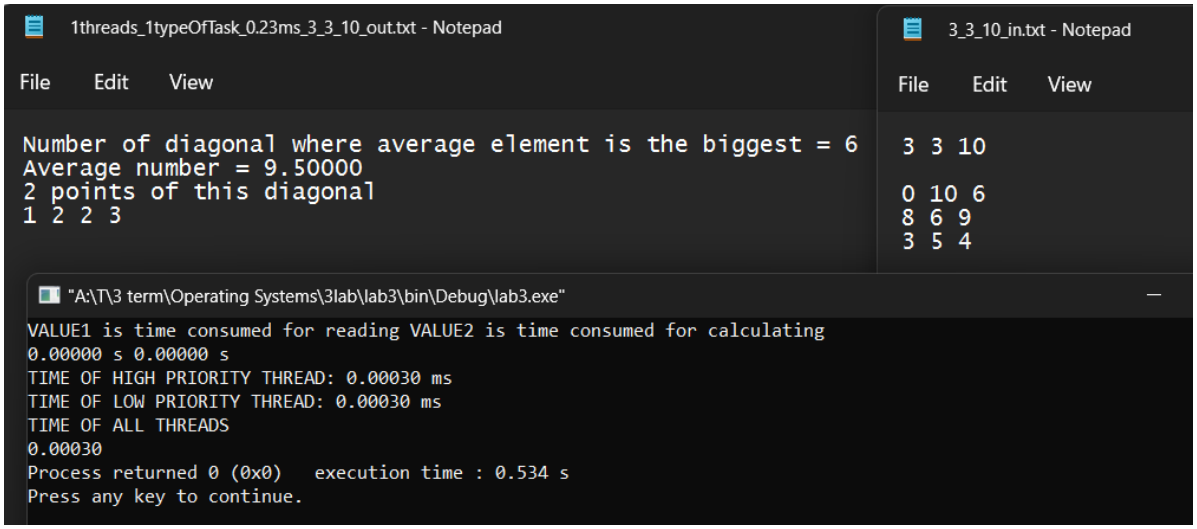
```
    ofstream output(result_path);

    //cout << process_time << " ms" << endl;
    output << "Number of diagonal where average element is the biggest = " << ans << " \n";
    output << fixed << setprecision(5) << "Average number = " << result << "\n";
    output << "2 points of this diagonal\n";
    output << points_for_diag[ans].first.first+1 << " " << points_for_diag[ans].first.second+1 << " " <<
points_for_diag[ans].second.first+1 << " " << points_for_diag[ans].second.second+1 << "\n";
    ld d1 = t1 - t0;
    ld d2 = t2 - t1;
    cout << "VALUE1 is time consumed for reading VALUE2 is time consumed for calculating" << endl;
    cout << fixed << setprecision(5) << d1/CLOCKS_PER_SEC << " s " << d2/CLOCKS_PER_SEC << " s"
<< endl;
    cout << "TIME OF HIGH PRIORITY THREAD: " << threadTime[high_priority_thread] << " ms" << endl;
    cout << "TIME OF LOW PRIORITY THREAD: " << threadTime[low_priority_thread] << " ms" << endl;
    cout << "TIME OF ALL THREADS" << endl;
    for(int i = 0; i < TOTAL_THREADS; ++ i){
        cout << threadTime[i] << " ";
    }
    return 0;
}
```

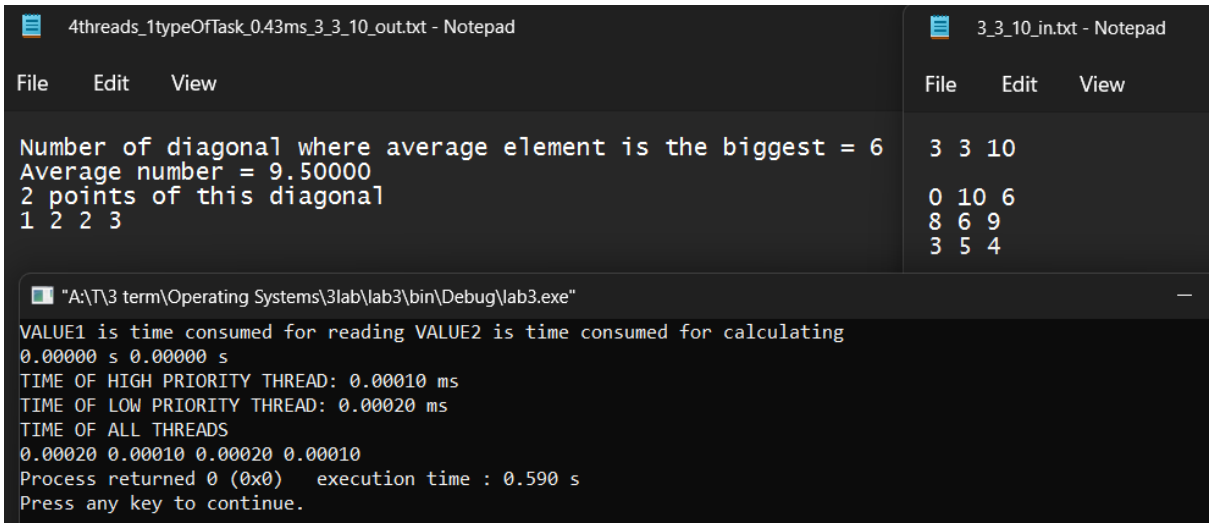
Вхідні - вихідні дані (+ Час виконання окремого потоку із пониженим/підвищеним пріоритетом у порівнянні із його «братами» стандартного пріоритету.):



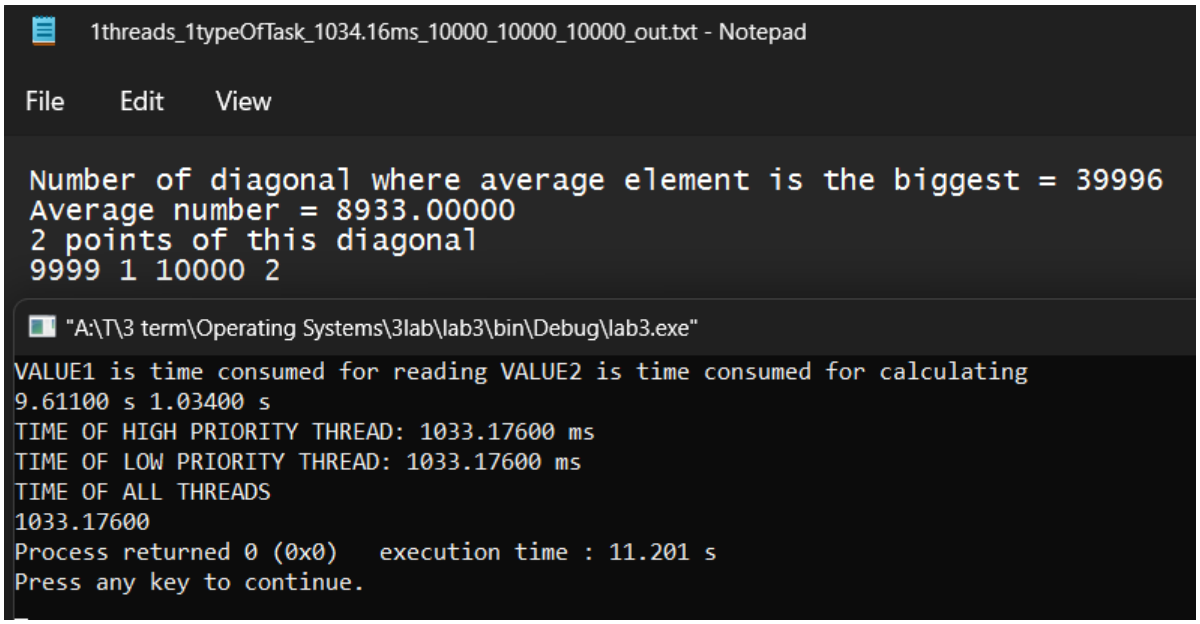


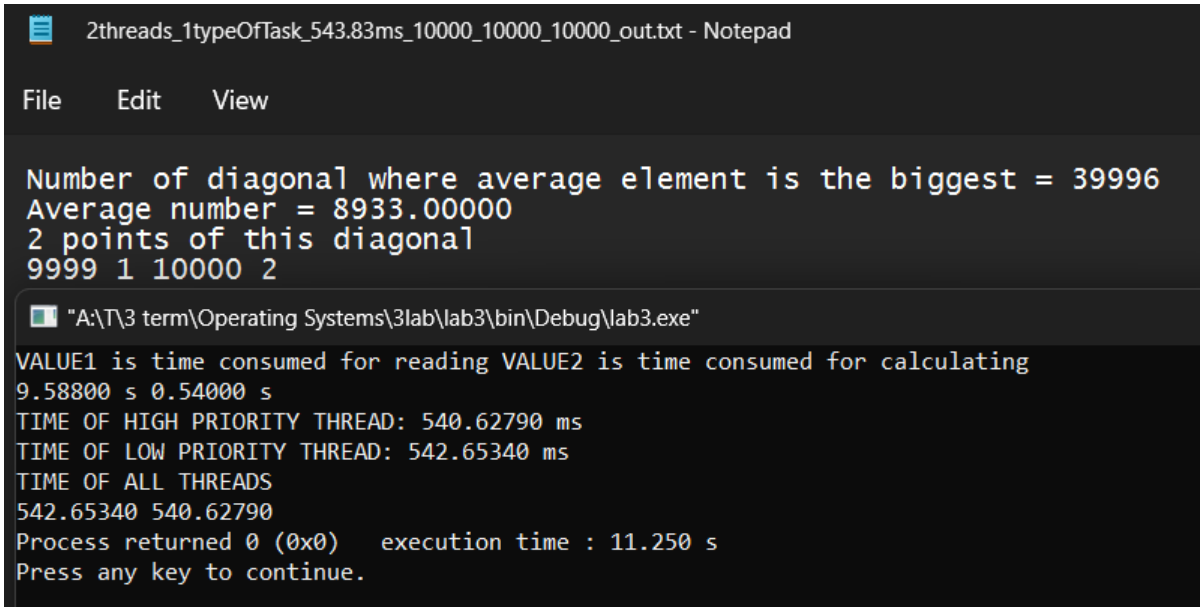
Next

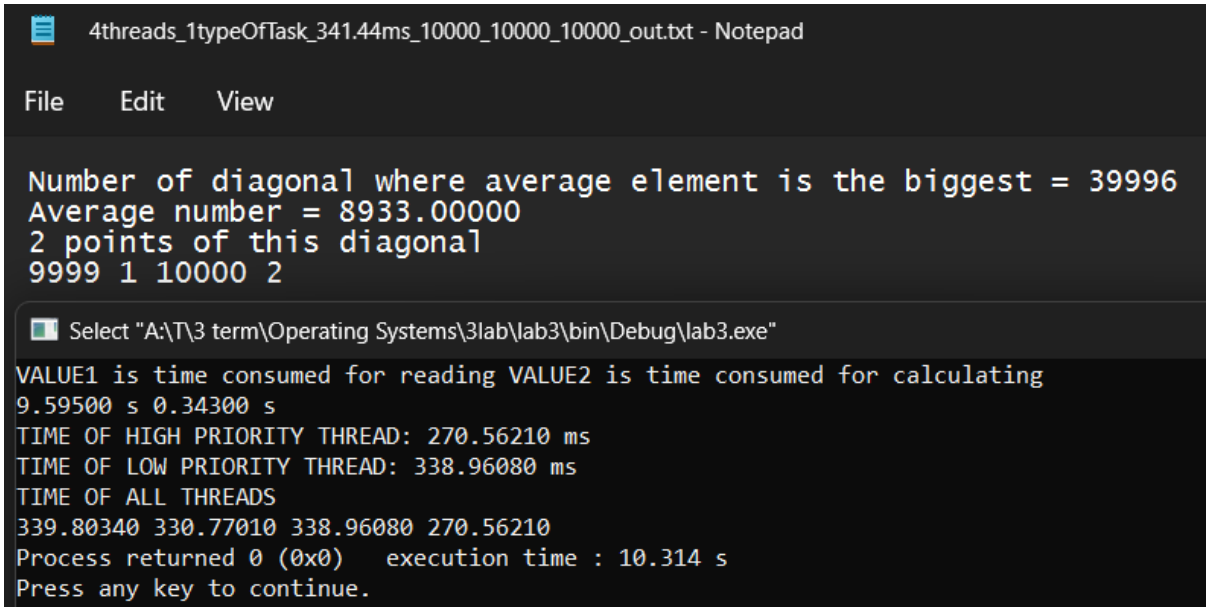


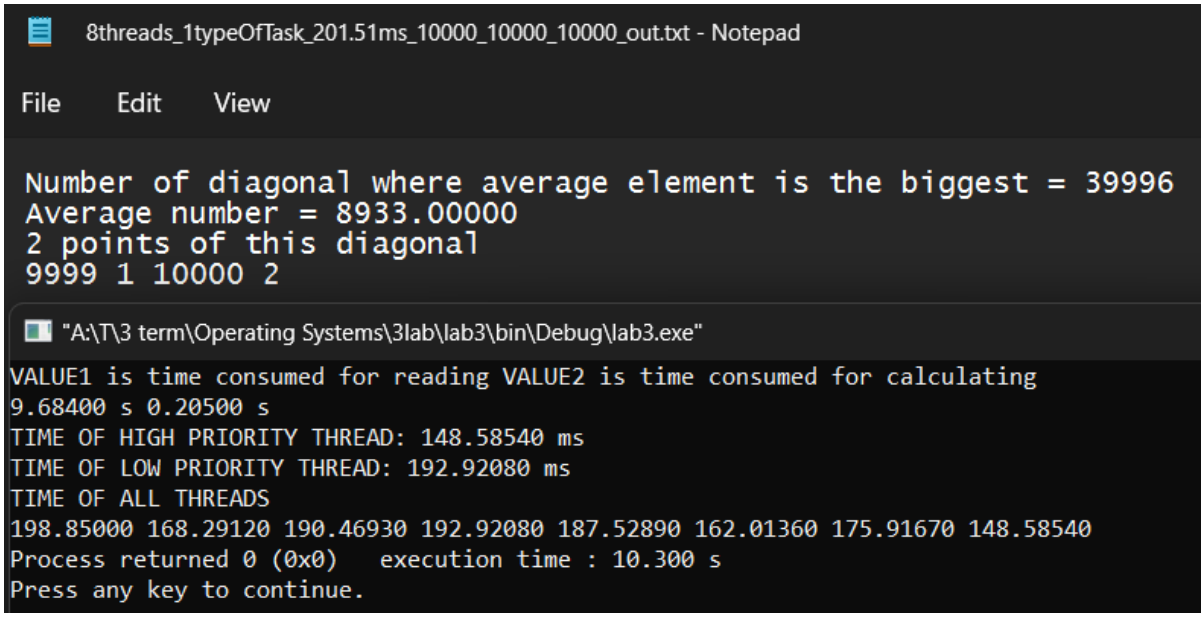


Next
https://drive.google.com/file/d/1AJXrMkF_wsElgnq6mCrg0gKZv0YwrIb3/view?usp=sharing
(Matrix 10000 x 10000).




2threads_1typeOfTask_543.83ms_10000_10000_10000_out.txt - Notepad
File Edit View
Number of diagonal where average element is the biggest = 39996
Average number = 8933.00000
2 points of this diagonal
9999 1 10000 2
"A:\T\3 term\Operating Systems\3lab\lab3\bin\Debug\lab3.exe"
VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
9.58800 s 0.54000 s
TIME OF HIGH PRIORITY THREAD: 540.62790 ms
TIME OF LOW PRIORITY THREAD: 542.65340 ms
TIME OF ALL THREADS
542.65340 540.62790
Process returned 0 (0x0) execution time : 11.250 s
Press any key to continue.


4threads_1typeOfTask_341.44ms_10000_10000_10000_out.txt - Notepad
File Edit View
Number of diagonal where average element is the biggest = 39996
Average number = 8933.00000
2 points of this diagonal
9999 1 10000 2
Select "A:\T\3 term\Operating Systems\3lab\lab3\bin\Debug\lab3.exe"
VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
9.59500 s 0.34300 s
TIME OF HIGH PRIORITY THREAD: 270.56210 ms
TIME OF LOW PRIORITY THREAD: 338.96080 ms
TIME OF ALL THREADS
339.80340 330.77010 338.96080 270.56210
Process returned 0 (0x0) execution time : 10.314 s
Press any key to continue.


8threads_1typeOfTask_201.51ms_10000_10000_10000_out.txt - Notepad
File Edit View
Number of diagonal where average element is the biggest = 39996
Average number = 8933.00000
2 points of this diagonal
9999 1 10000 2
"A:\T\3 term\Operating Systems\3lab\lab3\bin\Debug\lab3.exe"
VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
9.68400 s 0.20500 s
TIME OF HIGH PRIORITY THREAD: 148.58540 ms
TIME OF LOW PRIORITY THREAD: 192.92080 ms
TIME OF ALL THREADS
198.85000 168.29120 190.46930 192.92080 187.52890 162.01360 175.91670 148.58540
Process returned 0 (0x0) execution time : 10.300 s
Press any key to continue.

20threads_1typeOfTask_180.35ms_10000_10000_10000_out.txt - Notepad

File Edit View

```
Number of diagonal where average element is the biggest = 39996
Average number = 8933.00000
2 points of this diagonal
9999 1 10000 2
```

"A:\T\3 term\Operating Systems\3lab\lab3\bin\Debug\lab3.exe"

```
VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
9.55000 s 0.17900 s
TIME OF HIGH PRIORITY THREAD: 113.16810 ms
TIME OF LOW PRIORITY THREAD: 168.19880 ms
TIME OF ALL THREADS
136.90580 120.05580 110.83090 168.19880 113.61500 112.10770 110.79860 113.16810 130.52160 119.59250 147.79070 113.28720
113.51050 117.16860 117.72480 139.09160 113.95940 109.39230 70.91500 113.12650
Process returned 0 (0x0)   execution time : 10.320 s
Press any key to continue.
```

100threads_1typeOfTask_171.23ms_10000_10000_10000_out.txt - Notepad

File Edit View

```
Number of diagonal where average element is the biggest = 39996
Average number = 8933.00000
2 points of this diagonal
9999 1 10000 2
```

"A:\T\3 term\Operating Systems\3lab\lab3\bin\Debug\lab3.exe"

```
VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
9.63900 s 0.17000 s
TIME OF HIGH PRIORITY THREAD: 20.35140 ms
TIME OF LOW PRIORITY THREAD: 83.21560 ms
TIME OF ALL THREADS
20.96080 21.00920 20.73700 83.21560 20.05980 20.86240 21.68220 20.35140 22.32060 22.62850 22.14750 21.71970 31.50780 64.
58460 38.10490 22.65850 67.68340 45.63380 23.57690 75.72690 22.03110 21.57000 23.44190 22.40740 21.77920 22.80400 23.015
80 22.69560 22.11540 22.25020 24.84840 22.75410 20.79800 22.59030 22.96720 22.10660 23.00260 23.10720 46.61830 22.40810
25.91470 22.45550 22.95670 22.66070 48.59000 22.72670 22.13480 22.20230 28.05950 22.85700 22.48500 22.32120 23.04910 44.
24070 46.98980 21.58120 22.80970 19.92280 20.45240 21.62630 22.25740 23.19290 22.71720 22.04460 17.82860 43.24930 19.868
20 45.26900 21.02770 18.21000 45.16190 20.34670 45.25850 45.86240 44.90440 44.83610 45.80270 37.26230 46.66080 45.36910
22.57640 45.40490 25.43070 36.08230 22.91320 38.04380 22.37670 22.95590 23.22390 26.16610 24.95870 22.23480 23.01770 22.
08130 22.24450 22.55300 15.97560 25.45860 22.54750 23.07780
Process returned 0 (0x0)   execution time : 10.235 s
Press any key to continue.
```

1000threads_1typeOfTask_588.88ms_10000_10000_10000_out.txt - Notepad

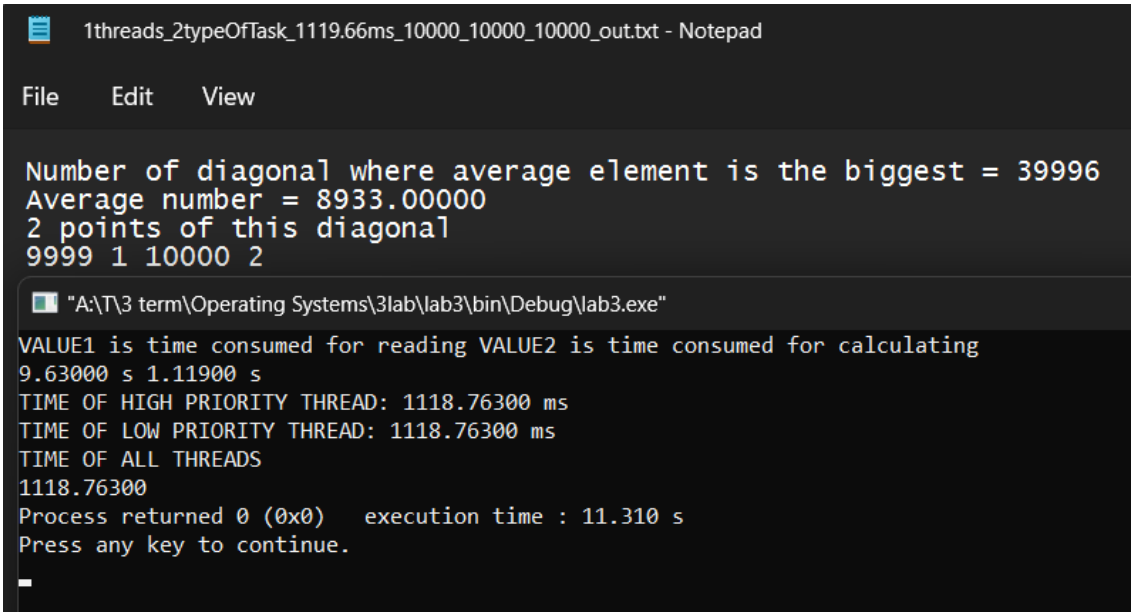
File Edit View

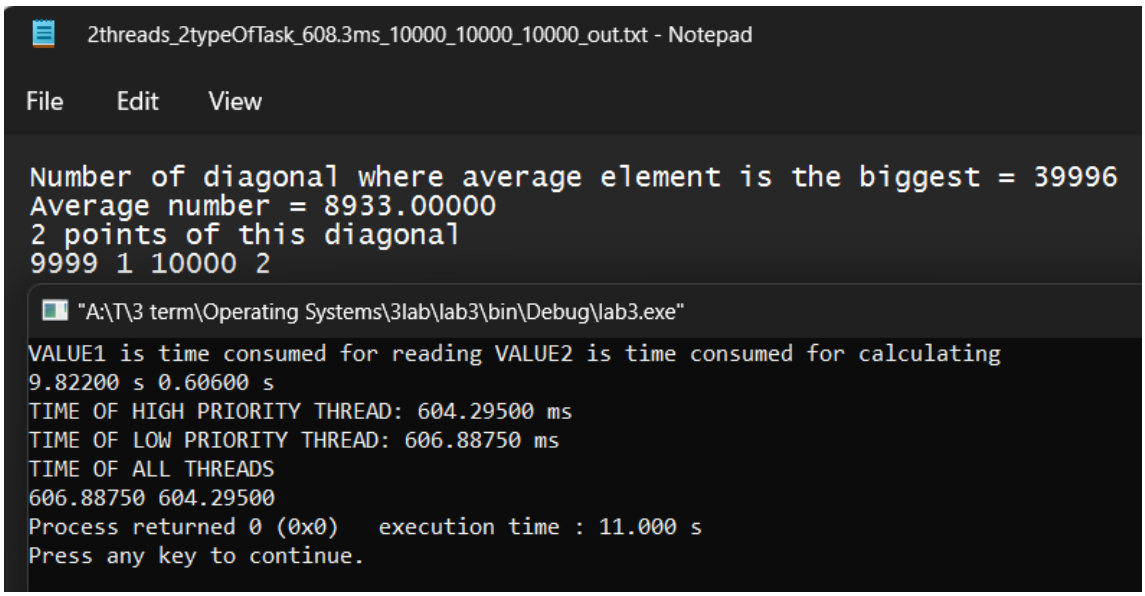
```
Number of diagonal where average element is the biggest = 39996
Average number = 8933.00000
2 points of this diagonal
9999 1 10000 2
```

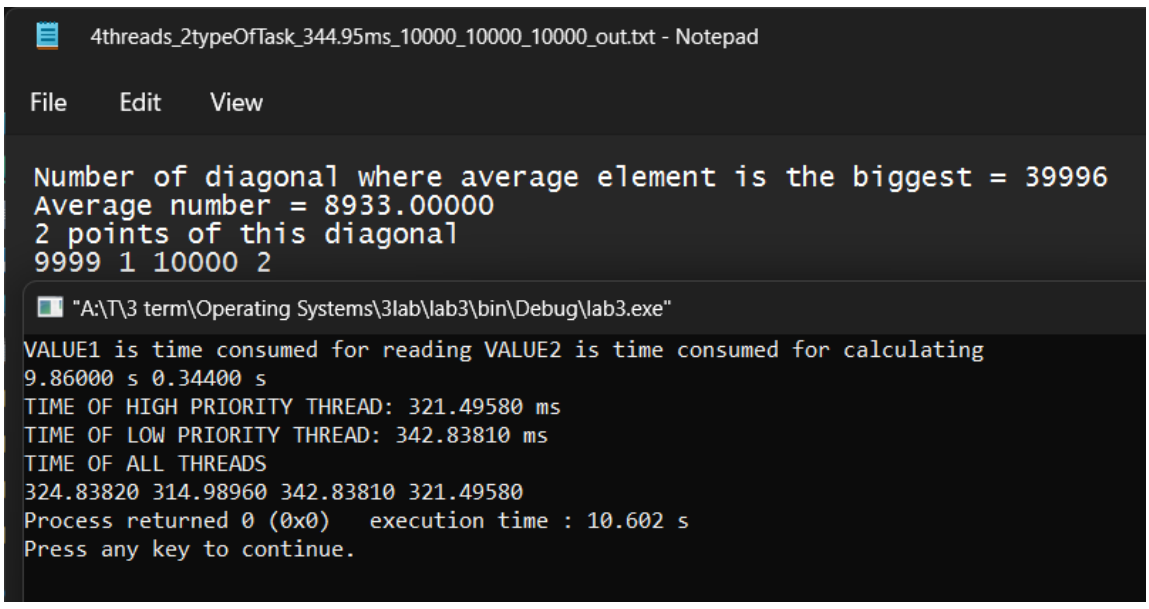
"A:\T\3 term\Operating Systems\3lab\lab3\bin\Debug\lab3.exe"

```
VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
9.77000 s 0.58900 s
TIME OF HIGH PRIORITY THREAD: 1.44340 ms
TIME OF LOW PRIORITY THREAD: 1.51070 ms
TIME OF ALL THREADS
1.16570 1.35670 1.48750 1.51070 1.56080 1.54050 1.54500 1.44340 1.38460 1.38500 1.43970 1.33030 1.34960 1.30200 1.33540 1.1
0 1.39220 1.33400 1.30270 1.30830 1.29340 1.25600 1.26680 1.28060 1.28020 1.25930 1.28770 1.26320 1.19740 1.14100 1.16240 1
180 1.17670 1.19640 1.11390 1.10240 1.16740 1.16700 1.16940 1.12870 1.10640 1.12400 1.12690 1.13240 1.13390 1.11750 1.1324
15240 1.17400 1.16110 1.13220 1.13100 2.07300 1.10930 1.13650 1.13110 1.12640 1.10250 1.14460 1.17090 1.14330 1.10020 1.12
1.14390 1.07620 1.12190 1.16100 1.15860 1.18500 1.19600 1.19550 1.29520 1.28600 1.18650 1.13350 1.13210 1.16970 1.15470 1.1
0 1.24230 1.25740 1.18820 1.23080 1.19170 1.17720 1.14310 1.12370 1.15880 1.15310 1.14660 1.10310 1.10520 1.21430 1.77150 1
630 1.73470 1.15100 1.09400 1.11020 1.12170 1.11060 1.12590 1.15250 1.13270 1.13490 1.12110 1.12530 1.11450 1.12180 1.1447
11170 1.14990 1.18420 1.16090 1.10840 1.11600 1.14240 1.13430 1.09850 1.10470 1.13350 1.10560 1.11080 1.14060 1.15750 1.15
1.12000 1.11670 1.10380 1.17240 1.21660 1.16080 1.12220 1.13210 1.16740 1.46840 1.33980 1.34470 1.13770 1.17730 1.16200 1.1
0 1.13620 1.18420 1.19670 1.22290 1.19040 1.11970 1.13620 1.12690 1.11440 1.14500 1.16830 1.21240 1.20290 1.60470 1.16150 1
770 1.13250 1.15390 1.18460 1.18190 1.14890 1.17820 1.10500 1.12350 1.13670 1.13130 1.09890 1.12860 1.12390 1.12890 1.1052
12700 1.12420 1.13260 1.11310 1.13340 1.13740 1.12650 1.10840 1.25780 1.17300 1.25760 1.25660 1.13110 1.12520 1.10580 1.12
1.13250 1.15400 1.14800 1.10350 1.12060 1.14860 1.10020 1.10040 1.10400 1.10430 1.11370 1.15560 1.15880 1.15540 1.11370 1.1
0 1.15420 1.13680 1.11780 1.12680 1.13160 1.13830 1.17750 1.20140 1.16690 1.16590 1.10970 1.22020 1.39030 1.59720 1.37100 1
220 1.17540 1.30850 1.18110 1.13540 1.11410 1.11510 1.09780 1.09950 1.20970 1.12870 1.09800 1.10300 1.11350 1.14650 1.1060
```

Те ж саме тільки не з граничним поділом(кожен TOTAL_TREADS`овий)







8threads_2typeOfTask_249.95ms_10000_10000_10000_out.txt - Notepad

File Edit View

```
Number of diagonal where average element is the biggest = 39996
Average number = 8933.00000
2 points of this diagonal
9999 1 10000 2
```

"A:\T\3 term\Operating Systems\3lab\lab3\bin\Debug\lab3.exe"

```
VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
9.79700 s 0.25000 s
TIME OF HIGH PRIORITY THREAD: 202.39430 ms
TIME OF LOW PRIORITY THREAD: 231.97220 ms
TIME OF ALL THREADS
243.19620 206.68300 239.25390 231.97220 223.21020 245.79640 223.88060 202.39430
Process returned 0 (0x0)   execution time : 10.472 s
Press any key to continue.
```

20threads_2typeOfTask_407.16ms_10000_10000_10000_out.txt - Notepad

File Edit View

```
Number of diagonal where average element is the biggest = 39996
Average number = 8933.00000
2 points of this diagonal
9999 1 10000 2
```

"A:\T\3 term\Operating Systems\3lab\lab3\bin\Debug\lab3.exe"

```
VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
9.59400 s 0.40800 s
TIME OF HIGH PRIORITY THREAD: 223.74100 ms
TIME OF LOW PRIORITY THREAD: 400.84870 ms
TIME OF ALL THREADS
285.31750 272.23040 284.82350 400.84870 224.59340 224.26500 224.05690 223.74100 296.70480 304.05660 264.29960 256.09210
246.57290 255.18360 266.58030 300.88750 284.75460 278.77360 259.34740 242.44120
Process returned 0 (0x0)   execution time : 15.079 s
Press any key to continue.
```

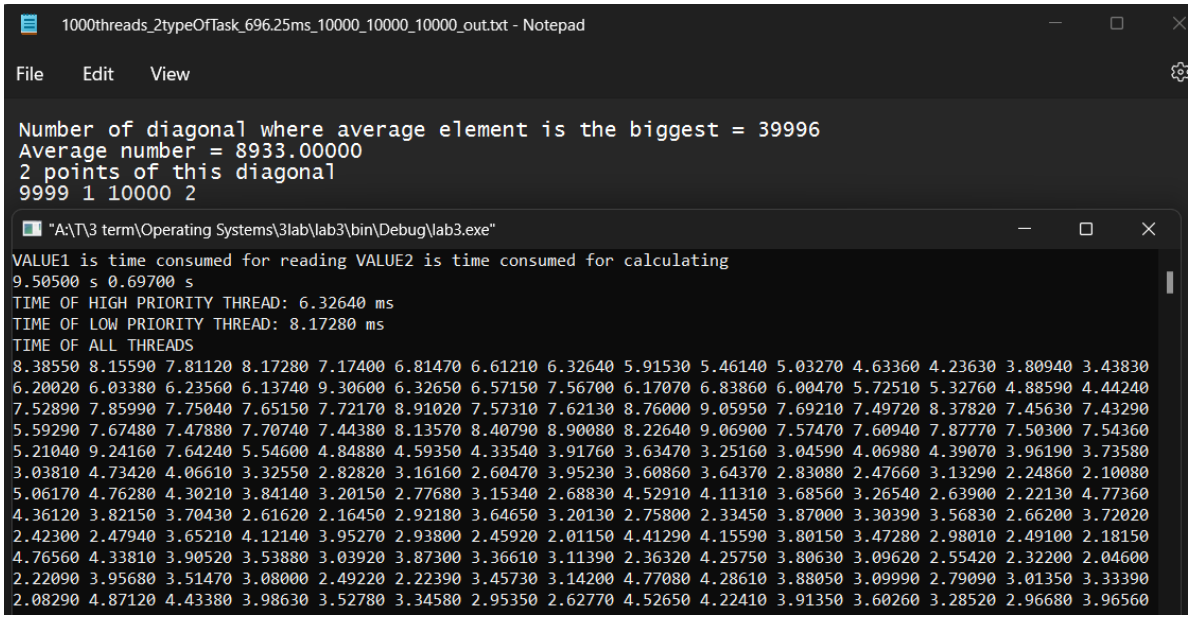
100threads_2typeOfTask_431.69ms_10000_10000_10000_out.txt - Notepad

File Edit View

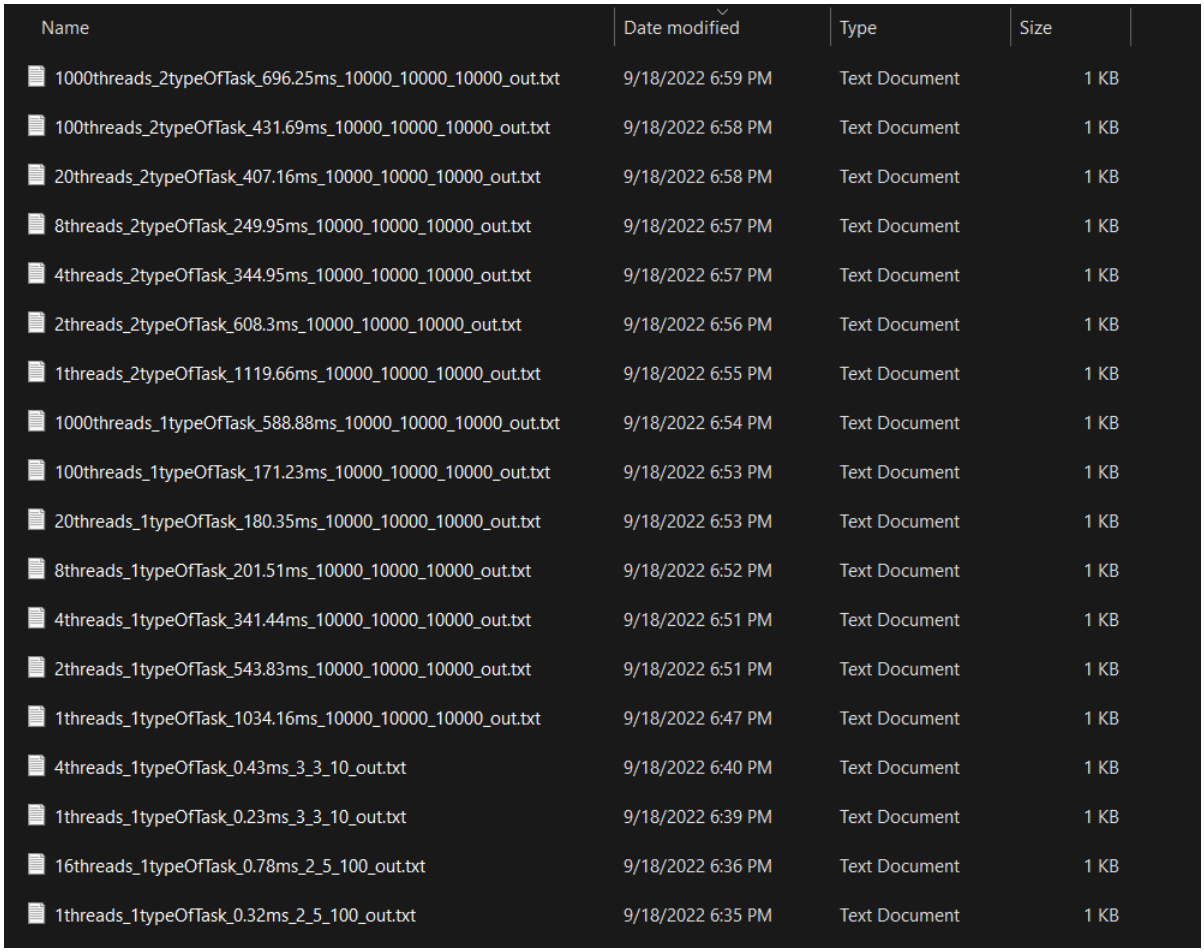
```
Number of diagonal where average element is the biggest = 39996
Average number = 8933.00000
2 points of this diagonal
9999 1 10000 2
```

"A:\T\3 term\Operating Systems\3lab\lab3\bin\Debug\lab3.exe"

```
VALUE1 is time consumed for reading VALUE2 is time consumed for calculating
9.60800 s 0.43100 s
TIME OF HIGH PRIORITY THREAD: 63.96680 ms
TIME OF LOW PRIORITY THREAD: 378.67630 ms
TIME OF ALL THREADS
127.96150 81.81210 107.87650 378.67630 64.93430 64.67710 93.59430 63.96680 248.21600 78.85010 62.54370 62.05840 103.7209
0 61.07010 133.60630 136.47410 83.71340 95.37620 183.79880 216.18350 139.26990 204.92610 204.90050 195.16920 164.76800 2
17.79090 239.10180 126.37680 131.37060 146.49230 161.51190 179.99880 164.66310 229.76960 203.13140 131.34740 187.38550 9
8.89770 98.87060 127.90230 168.21850 195.80350 133.96080 72.55600 72.47480 192.09220 178.86760 136.25000 120.28280 130.6
8870 134.85140 82.78950 267.37710 104.67960 120.73230 142.68670 141.49580 128.62390 186.49800 113.69430 120.13600 95.456
50 181.11560 124.12020 62.28620 126.57440 125.04450 61.18380 60.92400 60.50190 60.23020 111.62300 79.39850 58.86810 63.5
8150 84.15680 62.70320 62.27460 61.82410 0.00000 108.05590 91.49590 0.00000 93.80460 82.14450 92.47600 84.39190 90.76850
 0.00000 115.06000 110.50100 77.47270 77.44910 88.68550 82.39990 0.00000 0.00000 0.00000 93.86360 0.00000
```



Час виконання усіх прикладів з різною кількістю потоків і з різним розподілом:



Пояснення часових результатів:

Кількість потоків сильно впливає на затрачений час(При 20 потоках затрачений час в мому прикладі приблизно в 5 раз менший за затрачений час при 1 потоці).

Більше потоків – не завжди пришвидшення, на 1000 потоків час більший за час на 20 потоках.

Від перестановки доданків сума не змінюється. Поділ даних підряд в мому випадку є трішки швидшим тільки через програмну реалізацію(в цьому поділі є операції ділення та остачі від ділення, в той час як в послідовному поділі даних цих операцій немає). Загалом час має бути +- однаковим.

Пріоритети потоків впливають на їх час виконання. Потік з вищим пріоритетом виконується як один з найшвидших, потік з нижчим пріорітетом навпаки, виконується як один з найповільніших.

Висновок:

Я закріпив вміння та навички роботи з розпаралелюванням алгоритмів за допомогою багатопоточності в ОС Windows з використанням функцій WinAPI. Підготував файли з тестовими даними різного розміру. Вирішив поставлену задачу в окремому потоці. Результат зберіг у файл. Виконав розпаралелювання заданого алгоритму на 2, 4, 8, 20, 100, 1000 потоків. Реалізував можливість зміни пріоритету окремого потоку та проаналізував вплив на тривалість виконання його частини задачі.