#include <iostream>

#include <windows.h>

#include <chrono>

using namespace std;

using namespace std::chrono;

typedef struct SORT\_PARAMS {double \*time;SORT\_PARAMS(double \*t) : time(t) {}}

THREAD\_PARAMS;

const int N = 10;

int arr[N];

HANDLE mutex = CreateMutexA(nullptr, FALSE, nullptr);

DWORD WINAPI Thread(LPVOID lpParam)

{

auto start = high\_resolution\_clock::now();

int i, key, j;

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

{

WaitForSingleObject(mutex, INFINITE);

key = arr[i];

j = i - 1;

while (j >= 0 && arr[j] < key) {

arr[j + 1] = arr[j];

j = j - 1;

}

arr[j + 1] = key;

ReleaseMutex(mutex);

}

auto stop = high\_resolution\_clock::now();

double \*time = ((THREAD\_PARAMS \*) lpParam)->time, buftime;

\*time = double(duration\_cast<microseconds>(stop - start).count()) / 1000000;

::InterlockedExchangePointer(reinterpret\_cast<void \*volatile \*>(&buftime),

reinterpret\_cast<void

\*>(duration\_cast<microseconds>(stop - start).count() / 1000000));

}

int main()

{

cout << "=== An Array Top-Down Insert Sort ===" << endl;

char \*p;

char foo[2];

int size = 0;

for (int &i : arr)

{

i = rand()% 9000 + 1000;

for(p = itoa(i, foo, 10); \*p; p++, size++);

}

size += N;

int count;

cout << "Enter count of threads: ";

cin >> count;

double times[count];

size += count\*9;

cout << "Sorting...Please wait!" << endl;

HANDLE threads[count];

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

{

threads[i] = CreateThread(nullptr, 0, Thread, new

SORT\_PARAMS(&times[i]), 0, nullptr);

}

WaitForMultipleObjects(DWORD(count), threads, true, INFINITE);

for (HANDLE i: threads) CloseHandle(i);

HANDLE hFile, hMap;

LPBYTE pData = nullptr;

hFile = CreateFile(R"(f:\Programs\CLion\Projects\OS\prob\result.txt)",

GENERIC\_WRITE | GENERIC\_READ, 0, nullptr, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL,

NULL);

if (hFile != INVALID\_HANDLE\_VALUE)

{

hMap = CreateFileMapping(hFile, nullptr, PAGE\_READWRITE, 0, size,

nullptr);

if (hMap)

{

DWORD dwSize = GetFileSize(hFile, nullptr);

CloseHandle(hFile);

pData = (LPBYTE)MapViewOfFile(hMap, FILE\_MAP\_WRITE, 0, 0, 0);

if (pData)

{

int offset = 0;

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

{

for (p = itoa(arr[i], foo, 10); \*p; p++, offset++)

\*(pData + offset) = static\_cast<BYTE>(\*p);

\*(pData + offset) = '\t';

}

for(int i = 0; i < count; i++,offset++)

{

\*(pData + offset) = static\_cast<BYTE>(times[i]);

\*(pData + offset) = '\t';

}

UnmapViewOfFile(pData);

}

CloseHandle(hMap);

}

}

return 0;

}