**Системное программное обеспечение**

**Лабораторная работа №5**

Съестов Дмитрий Вячеславович

P3217

#include "stdafx.h"

#include <Windows.h>

#include <iostream>

using namespace std;

#define ARRAY\_LENGTH 10000

#define BUFFER\_SIZE 8

typedef enum {ASC, DESC} order\_t;

typedef struct {int\* arr; order\_t order; int x;} args\_t;

int cmp(int a, int b)

{

if (a > b) return 1;

if (a < b) return -1;

return 0;

}

void sort(int\* arr, size\_t n, order\_t order)

{

int gap, i, j, temp, ord = (order == ASC ? 1 : -1);

for (int gap = n/2; gap > 0; gap /= 2)

{

for (int i = gap; i < n; i += 1)

{

temp = arr[i];

j = i;

while (j >= gap && cmp(arr[j - gap], temp) == ord)

{

arr[j] = arr[j - gap];

j -= gap;

}

arr[j] = temp;

}

}

}

int bisect(int\* arr, size\_t n, order\_t order, int x)

{

int leftCmp = (order == ASC ? -1 : 1);

int first = 0, last = n;

while (first < last)

{

int i = (first + last) / 2;

if (x == arr[i]) return i;

else if (cmp(x, arr[i]) == leftCmp) last = i;

else first = i + 1;

}

return -1;

}

time\_t threadTime(HANDLE hThread)

{

FILETIME creationTime, exitTime, kernelTime, userTime;

ZeroMemory(&creationTime, sizeof(creationTime));

ZeroMemory(&exitTime, sizeof(exitTime));

GetThreadTimes(hThread, &creationTime, &exitTime, &kernelTime, &userTime);

time\_t startTime = ((time\_t)creationTime.dwHighDateTime << 32) + creationTime.dwLowDateTime;

time\_t endTime = ((time\_t)exitTime.dwHighDateTime << 32) + exitTime.dwLowDateTime;

return (endTime - startTime) / 10;

}

DWORD WINAPI ThreadSort(LPVOID lpParam)

{

args\_t args = \*(args\_t\*)lpParam;

sort(args.arr, ARRAY\_LENGTH, args.order);

bisect(args.arr, ARRAY\_LENGTH, args.order, args.x);

return EXIT\_SUCCESS;

}

int\* readArray()

{

char buffer[BUFFER\_SIZE];

int arr[ARRAY\_LENGTH];

FILE\* file = fopen("..\\array.txt", "r");

for (int i = 0; i < ARRAY\_LENGTH; i++)

{

if (fgets(buffer, BUFFER\_SIZE, file) == nullptr) break;

arr[i] = atoi(buffer);

}

fclose(file);

return arr;

}

int main(int argc, char\* argv[])

{

if (argc != 3 || (string)argv[1] != "asc" && (string)argv[1] != "desc")

{

cout << "Usage: " << endl << "spo4 {asc|desc} x" << endl;

system("pause");

return EXIT\_FAILURE;

}

order\_t order = (string)argv[1] == "asc" ? ASC : DESC;

int x = atoi(argv[2]);

int\* arr = readArray();

args\_t\* args[3];

for (int i = 0; i < 3; i++)

{

args[i] = (args\_t\*) malloc(sizeof(args\_t));

args[i]->arr = (int\*) malloc(ARRAY\_LENGTH \* sizeof(int));

memcpy(args[i]->arr, arr, ARRAY\_LENGTH \* sizeof(int));

args[i]->order = order;

args[i]->x = x;

}

HANDLE hThread[3];

DWORD dwThreadId[3];

for (int i = 0; i < 3; i++)

{

hThread[i] = CreateThread(NULL, 0, ThreadSort, args[i], 0, &dwThreadId[i]);

if (hThread[i] == NULL)

{

cout << "CreateThread failed: " << GetLastError() << endl;

system("pause");

return EXIT\_FAILURE;

}

}

WaitForMultipleObjects(3, hThread, TRUE, INFINITE);

DWORD handleCount;

GetProcessHandleCount(GetCurrentProcess(), &handleCount);

cout << "+------+----------------+" << endl <<

"| ID | TIME |" << endl <<

"+------+----------------+" << endl;

for(int i = 0; i < 3; i++)

{

time\_t time = threadTime(hThread[i]);

printf("|%6u|%12lli mcs|\n", dwThreadId[i], time);

CloseHandle(hThread[i]);

free(args[i]->arr);

free(args[i]);

}

cout << "+-------+----------------+" << endl << endl;

cout << "Handles count: " << handleCount << endl << endl;

system("pause");

}