

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук. Департамент программной инженерии.
Дисциплина: «Архитектура вычислительных систем»

Вариант 12. Определить индексы i, j , для которых существует наиболее длинная последовательность $A[i] < A[i+1] < A[i+2] < A[i+3] < \dots < A[j]$.

Входные данные: массив чисел A , произвольной длины большей 1000.

Количество потоков является входным параметром. Разработать программу с применением OpenMP.

Пояснительная записка

Выполнила: Кириченко Виктория
Фамилия Имя, Кириченко Виктория
студент гр. БПИ198.

Москва
2020

Содержание

| | |
|--|---|
| 1. Текст задания | 2 |
| 2. Применяемые расчетные методы | 2 |
| 2.1. Теория решения задания..... | 2 |
| 2.2. Дополнительный функционал программы | 2 |
| 3. Тестирование программы | 2 |
| 3.1. Корректные значения | 2 |
| 3.2. Некорректные значения | 4 |
| ПРИЛОЖЕНИЕ 1. Список литературы | 6 |
| ПРИЛОЖЕНИЕ 2. Код программы..... | 7 |

1. Текст задания

12. Определить индексы i, j , для которых существует наиболее длинная последовательность $A[i] < A[i+1] < A[i+2] < A[i+3] < \dots < A[j]$. Входные данные: массив чисел A , произвольной длины большей 1000. Количество потоков является входным параметром. Разработать программу с применением OpenMP.

2. Применяемые расчетные методы

2.1. Теория решения задания

Использовался итеративный параллелизм.

Источники информации:

1. Параллельное программирование на OpenMP. [Электронный ресурс] // URL: <http://ccfit.nsu.ru/arom/data/openmp.pdf> (дата обращения: 25.11.2020)
2. Race conditions and deadlocks. [Электронный ресурс] // URL: <https://docs.microsoft.com/en-us/troubleshoot/dotnet/visual-basic/race-conditions-deadlocks> (дата обращения: 26.11.2020)
3. Introduction to the OpenMP with C++ and some integrals approximation. [Электронный ресурс] // URL: <https://medium.com/swlh/introduction-to-the-openmp-with-c-and-some-integrals-approximation-a7f03e9ebb65> (дата обращения: 26.11.2020)

2.2. Дополнительный функционал программы

Помимо индексов i и j , программа выводит максимальную длину наибольшей возрастающей последовательности, а также время работы программы.

3. Тестирование программы

3.1. Корректные значения

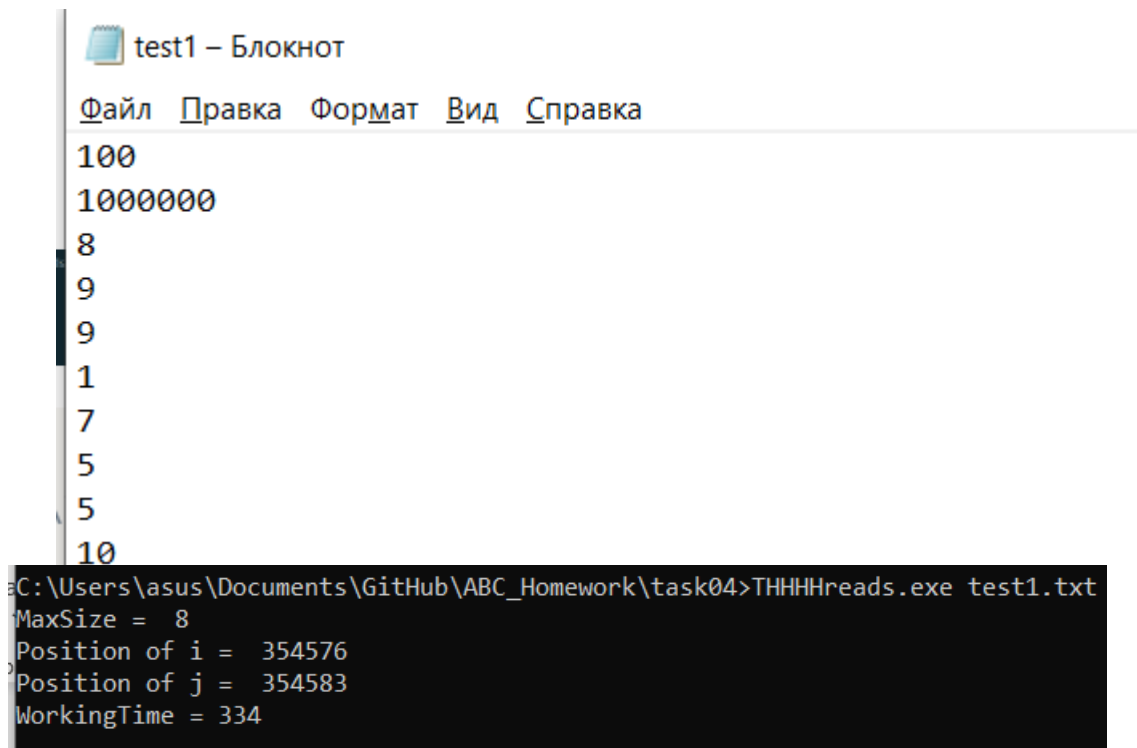
Для тестирования используется командная строка. 1 аргумент – exe файл программы, второй аргумент – путь к тестовому текстовому файлу (указывается либо полный путь, либо имя файла, в случае если файл лежит в той же папке, что и exe файл).



```
test1 - Блокнот
Файл Правка Формат Вид Справка
10
1000000
8
9
9
1
7
5
5
10
1
0
7
7
5
8
6
7
3
7
9
2
7
7
8
10
6
7
0

C:\Users\asus\Documents\GitHub\ABC_Homework\task04>TNNHreads.exe test1.txt
MaxSize = 8
Position of i = 354576
Position of j = 354583
WorkingTime = 53
```

1. Рисунок 1. Входные данные корректны



```

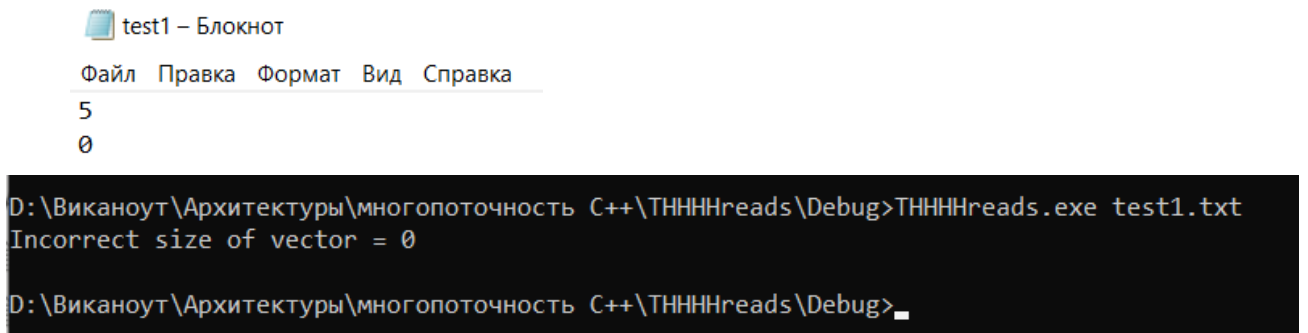
test1 – Блокнот
Файл  П_равка  Формат  В_ид  С_правка
100
1000000
8
9
9
1
7
5
5
10

C:\Users\asus\Documents\GitHub\ABC_Homework\task04>TNNHreads.exe test1.txt
MaxSize = 8
Position of i = 354576
Position of j = 354583
WorkingTime = 334

```

2. Рисунок 2. Входные данные корректны

3.2. Некорректные значения



```

test1 – Блокнот
Файл  П_равка  Формат  В_ид  С_правка
5
0

D:\Виканоут\Архитектуры\многопоточность C++\TNNHreads\Debug>TNNHreads.exe test1.txt
Incorrect size of vector = 0

D:\Виканоут\Архитектуры\многопоточность C++\TNNHreads\Debug>

```

3. Рисунок 3. Обработка некорректного размера массива (<1000)

test1 – Блокнот

Файл Правка Формат Вид Справка

5

kjhvg

sdfcv

```
D:\Виканоут\Архитектуры\многопоточность C++\ТНННreads\Debug>ТНННreads.exe test1.txt
Incorrect file
```

```
D:\Виканоут\Архитектуры\многопоточность C++\ТНННreads\Debug>
```

4. Рисунок 4. Обработка некорректных данных

test1 – Блокнот

Файл Правка Формат Вид Справка

-5

1000000

8

9

9

1

7

5

5

10

1

0

7

7

```
D:\Виканоут\Архитектуры\многопоточность C++\ТНННreads\Debug>ТНННreads.exe test1.txt
Incorrect size of vector = 1000000 or incorrect threadNumber = -5
```

```
D:\Виканоут\Архитектуры\многопоточность C++\ТНННreads\Debug>
```

Рисунок 5. Обработка некорректного количества потоков

ПРИЛОЖЕНИЕ 1**Список литературы**

4. Заголовок. [Электронный ресурс] // URL: ссылка (дата обращения: дата)

Код программы

```

#include <iostream>
#include <fstream>
#include <iomanip>
#include <limits>
#include <ctime>
#include <thread>
#include <vector>
#include <mutex>
#include <string>
#include <omp.h>
using namespace std;

int main(int argc, char* argv[]) {
    string pathFrom = argv[1];
    string n;
    std::string line;
    string threadNum;
    std::ifstream in(pathFrom); // открываем файл для чтения
    string res;
    double* A = nullptr;
    int index = 0;
    int n1 = 0;
    int thrNum = 0;
    if (in.is_open()) {
        getline(in, threadNum);
        getline(in, n);

        try {
            thrNum = stoi(threadNum);
            n1 = stoi(n);
            A = new double[n1];
            while (getline(in, line)) { // считываем массив из файла
                if (index < n1) {
                    A[index] = stoi(line);
                    index++;
                }
            }
        }
        catch (int a) {
            std::cout << "We caught an int exception with value: " << a << '\n';
            return 1;
        }
        catch (const std::invalid_argument& ia) {
            std::cerr << "Incorrect file " << '\n';
            return 1;
        }
    }
    if (n1 < 1000 || thrNum < 1) {
        std::cout << "Incorrect size of vector = " << n1 << " or incorrect threadNumber = "
        << thrNum << "\n";
        return 1;
    }
    if (index != n1) {

```



```

        std::cout << "Incorrect size of vector = " << n1 << "\n";
        return 1;
    }
    clock_t start_time = clock();

    int MaxSize ;
    int pos;
    vector<int> MaxSizes;
    vector<int> poses;

#pragma omp parallel shared(A,MaxSize,pos,MaxSizes,poses,thrNum)
    {
#pragma omp for
        for (int i = 0; i < thrNum; i++)
        {
            int k = n1 / thrNum;
            int start = i * k;
            int finish = 0;
            if (i == thrNum - 1) {
                finish = n1 - 1;
            }
            else {
                finish = (i + 1) * k;
            }
            int d = 1;

            int* d1;
            d1 = new int[n1];

            for (int j = start; j < finish; ++j) {
                d1[j] = 1;
                int g = n1;
                for (int l = j + 1; l < g; ++l) {
                    if (A[l] > A[l - 1]) {
                        d1[j]++; //если след эл-т больше, то длина последовательности
увеличивается
                    }
                    else {
                        g = 0; //чтобы завершить цикл предварительно
                    }
                }
                if (d1[j] > MaxSize) {
                    MaxSize = d1[j];
                    pos = j;
                }
            }
        }
    }

    clock_t end_time = clock();

    std::cout << "MaxSize = " << MaxSize << "\n";
    std::cout << "Position of i = " << pos << "\n";
    std::cout << "Position of j = " << pos+MaxSize-1 << "\n";
    std::cout << "WorkingTime = " << end_time - start_time << "\n";
    delete[] A;
    return 0;
}

```