

**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.**

Пояснительная записка

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

## 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.

## 2. Применяемые расчетные методы

### 2.1. Теория решения задания

Использовался итеративный параллелизм.

Источники информации:

1. Параллельное программирование в модели параллелизм данных. [Электронный ресурс] // URL: [http://www.ccas.ru/paral/prog/data\\_par/prog.html](http://www.ccas.ru/paral/prog/data_par/prog.html) (дата обращения: 10.11.2020)
2. Многопоточность, конкурентность и параллелизм: основы. [Электронный ресурс] // URL: <https://medium.com/nuances-of-programming/c-%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C-%D0%BA%D0%BE%D0%BD%D0%BA%D1%83%D1%80%D0%B5%D0%BD%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C-%D0%B8-%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%BB%D0%B5%D0%BB%D0%B8%D0%B7%D0%BC-%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-86bfa8679aed> (дата обращения: 10.11.2020)

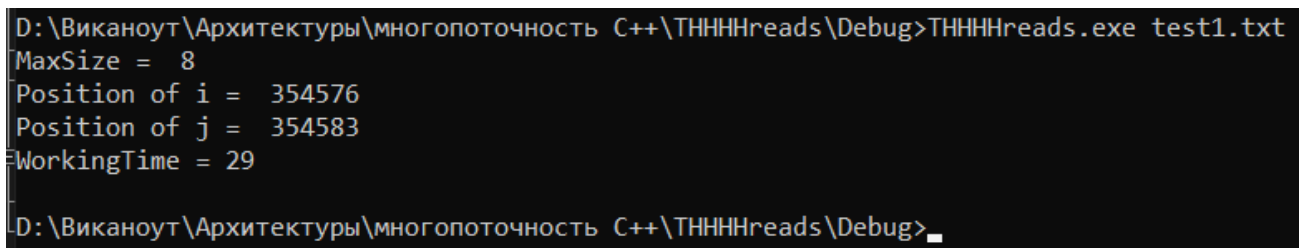
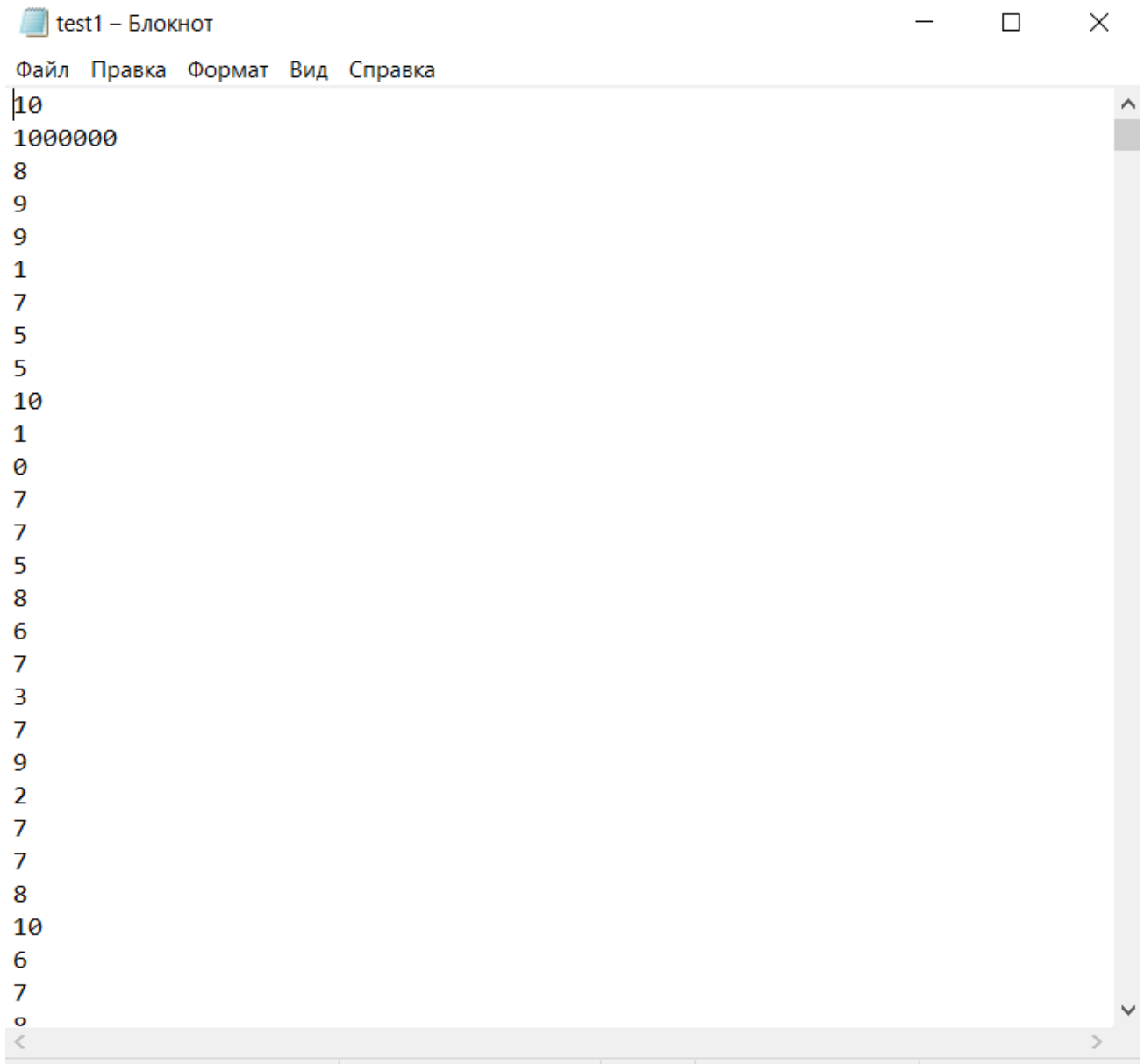
### 2.2. Дополнительный функционал программы

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

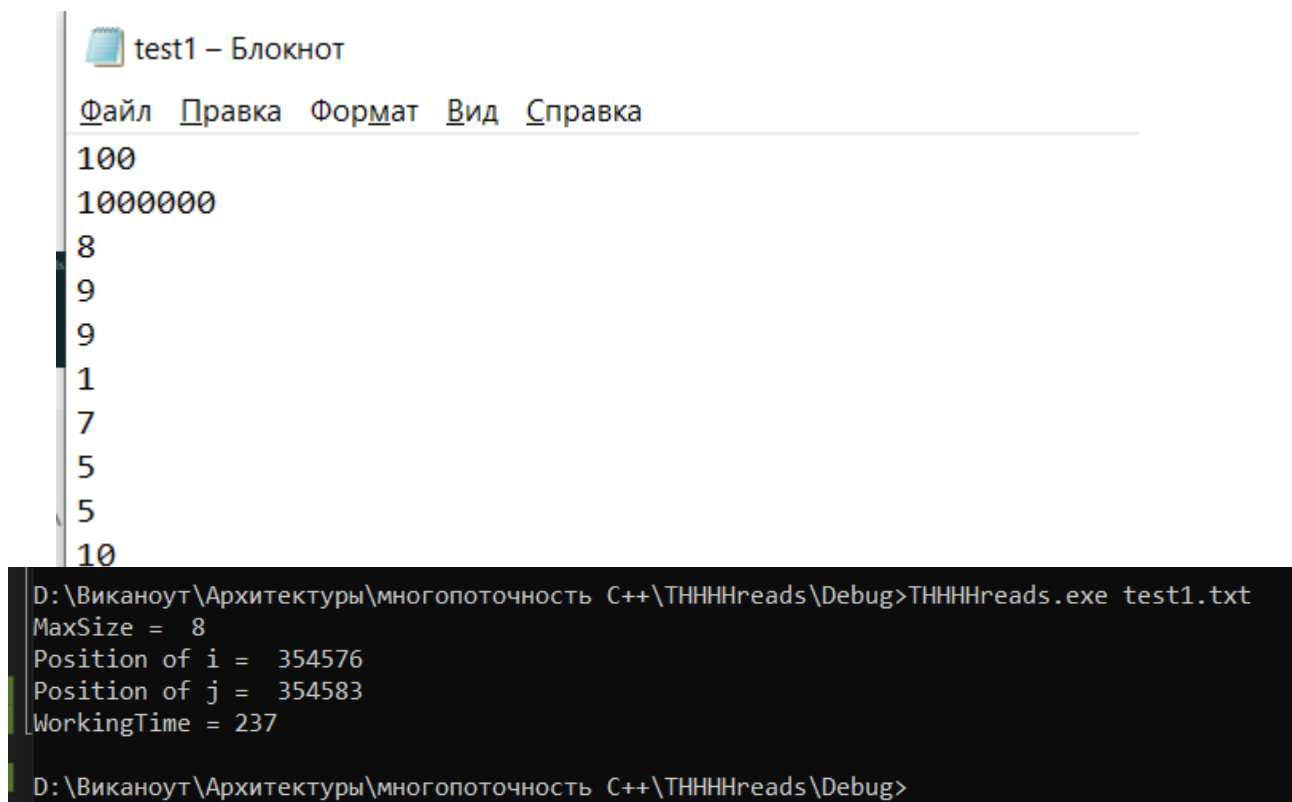
## 3. Тестирование программы

### 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).



1. Рисунок 1. Входные данные корректны



The image shows a Notepad window titled "test1 – Блокнот" with a menu bar (Файл, Правка, Формат, Вид, Справка) and the following text:

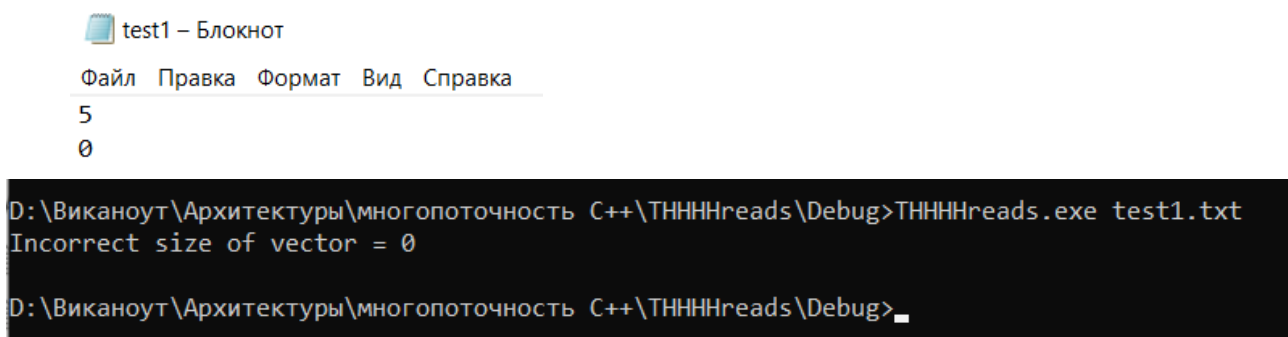
```
100
1000000
8
9
9
1
7
5
5
10
```

Below the Notepad window is a terminal window showing the execution of the program:

```
D:\Виканоут\Архитектуры\многопоточность C++\ТННННreads\Debug>ТННННreads.exe test1.txt
MaxSize = 8
Position of i = 354576
Position of j = 354583
WorkingTime = 237
D:\Виканоут\Архитектуры\многопоточность C++\ТННННreads\Debug>
```

2. Рисунок 2. Входные данные корректны

### 3.2. Некорректные значения



The image shows a Notepad window titled "test1 – Блокнот" with a menu bar (Файл, Правка, Формат, Вид, Справка) and the following text:

```
5
0
```

Below the Notepad window is a terminal window showing the execution of the program:

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

3. 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>
using namespace std;
std::mutex g_lock;

/// <summary>
/// метод, подсчитывающий макс длину возрастающей последовательности в пределах значений
/// </summary>
/// <param name="n">кол-во элементов массива</param>
/// <param name="a">массив</param>
/// <param name="start">верхний предел</param>
/// <param name="finish">нижний предел</param>
/// <param name="Max">максимальное кол-во эл-в в возрастающей последовательности</param>
/// <param name="position">позиция эл-та от которого начинается отсчет</param>
/// <param name="potok">номер потока</param>
void lap(int n, double* a, int start, int finish, double& Max, double& position, int potok)
{
    //к - начало , l - конец int& MaxSize, int& k, int& l, int start, int finish
    //g_lock.lock();
    int* d; //массив эл-ов из длин последовательностей начиная с i-ого эл-та
    d = new int[n]; // константа MAXN равна наибольшему возможному значению n
    for (int i = start; i < finish; ++i) {
        d[i] = 1;
        int k = n;
        for (int j = i + 1; j < k; ++j) {
            if (a[j] > a[j - 1]) {
                d[i]++; //если след эл-т больше, то длина последовательности увеличивается
            }
            else {
                k = 0; //чтобы завершить цикл предварительно
            }
        }
    }
    double MaxSize = 0;
    int pos = -1;
    for (int i = 0; i < n; i++) {
        if (d[i] > MaxSize) {
            MaxSize = d[i];
            pos = i;
        }
    }
    Max = MaxSize;
    position = pos;
    //cout << "potok№ " << potok << " MaxSize = " << MaxSize << " pos = " << pos << endl;
    //g_lock.unlock();
}

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 = new double[0];
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;
}

std::thread* thr = new thread[thrNum];
double* maxOfThreads = new double[thrNum];
double* positions = new double[thrNum];
clock_t start_time = clock();

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;
    }
    thr[i] = std::thread{ lap, n1, A, start, finish,
ref(maxOfThreads[i]), ref(positions[i]), i };
}

```

```

double MaxSize = 0.0;
int pos;
vector<int> MaxSizes;
vector<int> poses;
for (int i = 0; i < thrNum; i++) {
    thr[i].join();
    MaxSizes.push_back(maxOfThreads[i]);
    poses.push_back(positions[i]);
}
delete[] thr;
for (int i = 0; i < MaxSizes.size(); i++)
{
    if (MaxSizes[i] > MaxSize) {
        MaxSize = MaxSizes[i];
        pos = poses[i];
    }
}

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;
}

```