ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа бакалавриата «Программная инженерия»

Пояснительная записка к четвертому домашнему заданию по дисциплине “Архитектура вычислительных систем”

Преподаватель: Легалов Александр Иванович
Студент: Мкртумян Роберт Манвелович

Группа: ФКН БПИ194

Вариант: 12

Москва 2020

Текст задания

Определить индексы i, j, для которых существует наиболее длинная последовательность A[i] < A[i+1] < A[i+2] < A[i+3] < … < A[j]. Входные данные: массив чисел A, произвольной длины большей 1000. Количество потоков является входным параметром.

Используемая модель вычислений

При разработке использовалась итеративная модель построения многопоточных приложений, так как в данной задаче потоки равноправны и выполняют идентичные циклические задачи

Алгоритм работы программы

1) Получаем количество потоков
2) В параллельном цикле ищем для каждого элемента ряда максимальную длину неубывающего до него ряда
3) Ищем максимальную длину ряда
4) Записываем результат в файл

Ввод/Вывод

Входные данные передаются в качестве аргументов командной строки в следующем формате:

<путь до файла с входными данными>_<путь до выходного файла>_<число потоков>

В входном файле

На первой строке записана длина массива, далее в каждой строке каждый элемент

В выходном файле

На первой строке - длина последовательности.

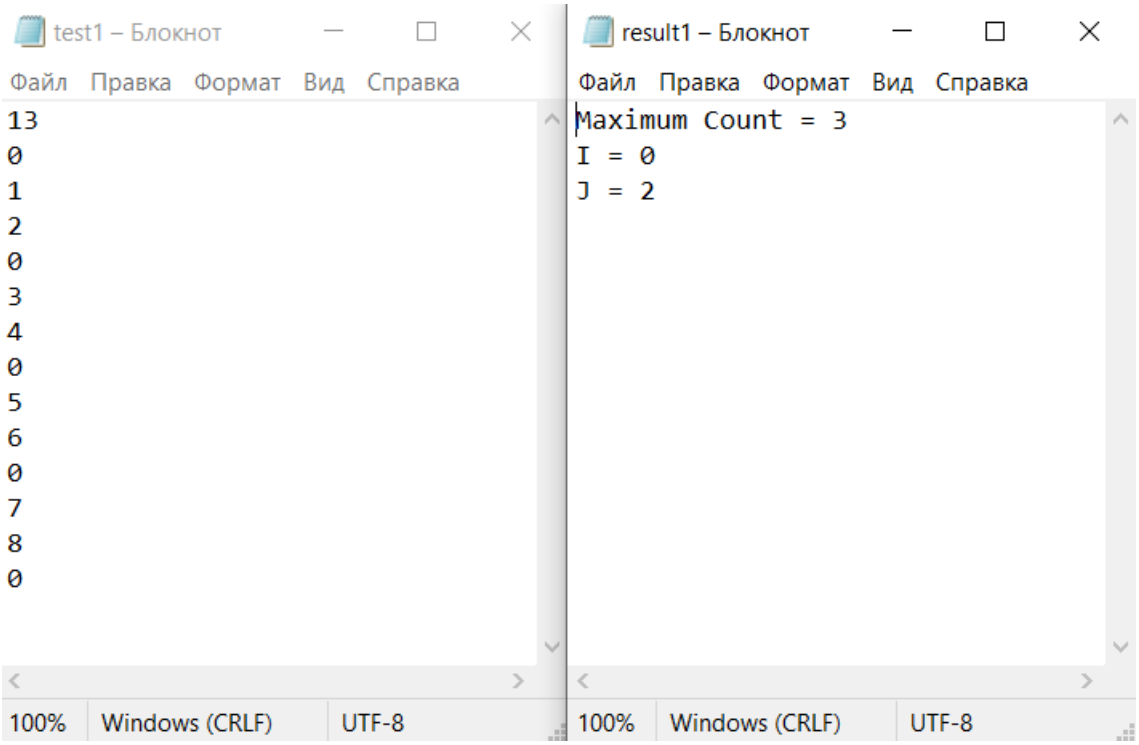
На второй строке - индекс начала последовательности.

На третьей строке - индекс конца последовательности.

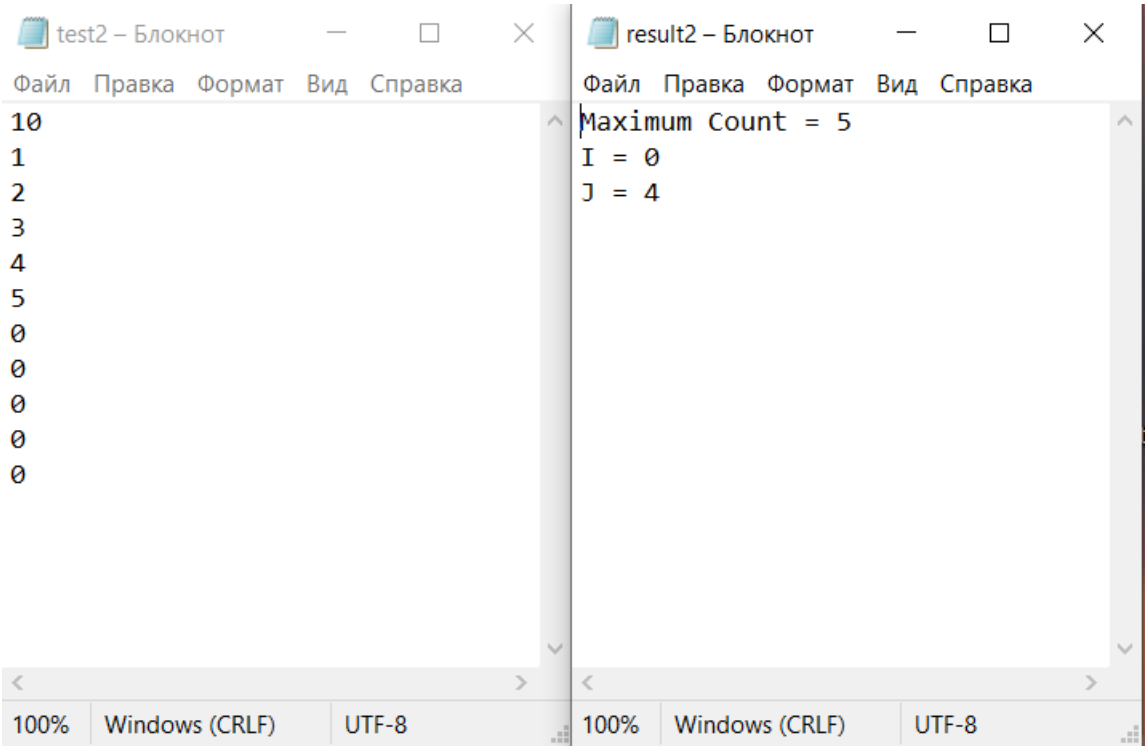
Тестирование

Тесты с массивами длины >1000 **можно найти** в репозитории в файле **test/input**.
Здесь для удобства рассмотрены тесты с небольшими входными данными.

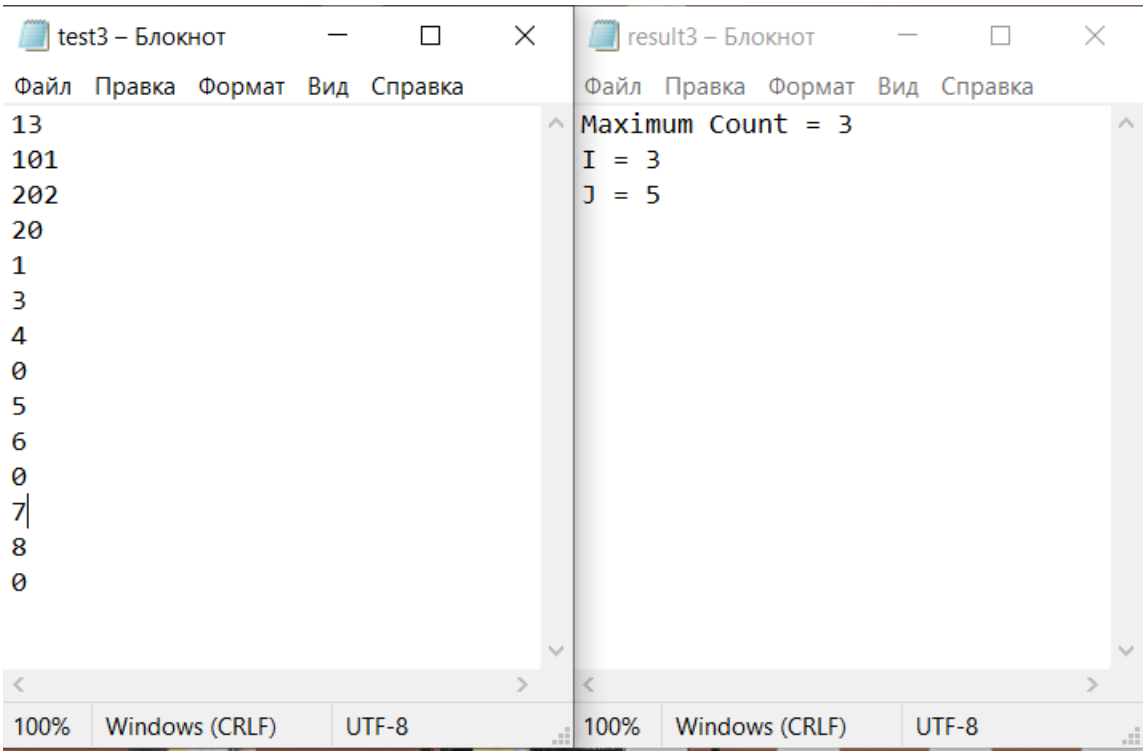
Тест 1



Тест 2



Тест 3



Список используемых источников

1) А.И. Легалов “Архитектура параллельных вычислительных систем. Многопотосность”
(http://www.softcraft.ru/edu/comparch/lect/07-parthread/)
2) А.И. Легалов “Многопоточное программирование. OpenMP”

(http://www.softcraft.ru/edu/comparch/practice/thread/03-openmp/)
3) Блог программиста: Библиотека OpenMP. Параллельные циклы (https://pro-prof.com/archives/1150#page_1)
4) Блог программиста: Учебник по OpenMP (https://pro-prof.com/archives/4335#more-4335)

Текст программы

```
#include <iostream>
#include <stdio.h>
#include <sstream>
#include <string>
#include <omp.h>
#include <fstream>
#include <vector>
#include <algorithm>

using namespace std;

int threadCount = 1;    // Количество потоков (по умолчанию программа
однопоточавая)
int arraySize = -1;
int* A;                 // Исходный числовой ряд
int* lenghts;         // Длина неубывающего ряда до конкретного элемента
int maxCount = -1;     // Максимальная длина ряда
int I = -1;
int J = -1;

int main(int argc, char* argv[])
{
   if (argc != 4) {    // Проверяем правильность входных аргументов
      cout << "Incorrect input data" << endl;
      return 0;
   }

   string in_file = argv[1];  // Путь до файла с входными данными
   string out_file = argv[2]; // Путь до выходного файла
   threadCount = stoi(string(argv[3]));

   ifstream fin;  // Поток чтения из файла
   fin.open(in_file);
   ofstream fout; // Поток записи в авходной файл
   fout.open(out_file);
   if (!fin.is_open() || !fout.is_open()) {
      cout << "File input/output error" << endl;
      return 0;
   }


   fin >> arraySize;      // Получаем длину ряда
   A = new int[arraySize];
   lenghts = new int[arraySize];
   for (size_t i = 0; i < arraySize; i++)
   {
      lenghts[i] = 1;
   }
   unsigned int index = 0;
   int num;
   while (!fin.eof() && index < arraySize)     // Заполняем массив чисел
   {
```

```
      fin >> num;
      A[index] = num;
      index++;
   }
   fin.close();    // Закрываем поток чтения



#pragma omp parallel num_threads(threadCount) // Создаем параллельную область
выполнения
   {
# pragma omp for // Распараллеливаем цикл
      for (int i = 0; i < arraySize; i++)
      {  // Ищем длину неубывающего ряда для каждого элемента
         for (int j = 0; j < i; j++)
         {
            if (A[j] < A[j+1] && j < arraySize)
            {
               lenghts[i] = lenghts[i] + 1;
            }
            else {
               lenghts[i] = 1;
            }
         }
      }
   }


   for (int i = 0; i < arraySize; i++)
   {  // Ищем максимальную длину ряда
      if (lenghts[i] > maxCount)
      {
         maxCount = lenghts[i];
         J = i;
         I = J - maxCount + 1;
      }
   }

   // Выводим информацию в файл
   fout << "Maximum Count = " << maxCount << "\n";
   fout << "I = " << I << "\n";
   fout << "J = " << J << "\n";

   cout << "Main end..." << endl;
}
```