ДСОГЛАСОВАНО

Научный руководитель,  
доцент департамента программной инженерии факультета компьютерных, канд. техн. наук

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.И. Легалов

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

УТВЕРЖДАЮ

**Домашняя работа**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.04.01-01 81 01-1 ЛУ**

Исполнитель

cтудент группы БПИ197  
\_\_\_\_\_\_\_\_\_\_\_\_\_/ П. О. Кулешова /

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Академический руководитель образовательной программы «Программная инженерия» профессор департамента программной инженерии, канд. техн. наук

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_В. В. Шилов

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020 г.

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл |  |

**Домашняя работа**

**Пояснительная записка**

**RU.17701729.04.01-01 81 01-1**

**Листов 13**

сОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc56548773)

[1.1. Наименование программы 3](#_Toc56548774)

[1.2. Документ, на основе которого ведется разработка 3](#_Toc56548775)

[1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc56548776)

[1.1 Назначение разработки 4](#_Toc56548777)

[1.2 Краткая характеристика области применения 4](#_Toc56548778)

[2. Описание программы 5](#_Toc56548779)

[Условие задачи 5](#_Toc56548780)

[Метод решения 5](#_Toc56548781)

[Алгоритм решения 5](#_Toc56548782)

[Замечание 5](#_Toc56548783)

[ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ 7](#_Toc56548784)

[ПРИЛОЖЕНИЕ 2 Код программы 8](#_Toc56548785)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 12](#_Toc56548786)

# ВВЕДЕНИЕ

## Наименование программы

Наименование программы: Практические приемы построения многопоточных приложений

Краткое наименование программы: ДЗ

## Документ, на основе которого ведется разработка

http://www.softcraft.ru/edu/comparch/tasks/t03/

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

## Назначение разработки

Разработать многопоточную программу для нахождения наибольшей длины

## Краткая характеристика области применения

Научно-исследовательская область.

# Описание программы

## Условие задачи

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

## Метод решения

Задача решалась с использованием **итеративного параллелизма.** Так как при поиске подпоследовательности мы не записываем никакие дополнительные данные и все потоки равноправны.

## Алгоритм решения

1. Разделяем массив на число промежутков равное числу потоков
2. Каждый поток проходится по своему промежутку и запоминает наибольшую длину возрастающей последовательности.
   1. Если к концу промежутка он обнаружил, что следующий элемент существует и он больше текущего, то он продолжает идти по последовательности до окончания возрастающей последовательности.
3. Когда все потоки закончили работу, то проходимся по всему массиву результатов и находим наилучший. Результат записываем в файл.

## Замечание

Считалось, что последовательность из одного символа является возрастающей. Искалась первая наибольшая последовательность.**3. Формат входных данных**

**Входные данные командной строки**

Программа на вход получает данные из входной строки в следующем формате: *<Путь до файла с тестом>\_<Путь до файла с ответом>\_<Число потоков>*

Значок **\_** означает пробел, то есть входные данные разделены одинарным пробелом.

**Входные данные для тестов**

Тест должен состоять из строки с числом элементов в массиве. На второй строке – элементы массива через пробел.

1. **формат выходных данных**

Текстовый файл, где на первой строке записан индекс начала последовательности, на второй – индекс окончания, на третьей – длина последовательности.

# ИСТОЧНИКИ, ИСПОЛЬЗОВАННЫЕ ПРИ РАЗРАБОТКЕ

1. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению// Единая система программной документации. – М.: ИПК Стандартинформ, 2010.
2. Сайт «SoftCraft». URL: http://www.softcraft.ru/edu/comparch/practice/thread/01-simple/ [http://www.softcraft.ru] Просмотрено: 17.11.2020
3. Сайт «Основы многопоточного, параллельного и распределенного программирования ГРЕГОРИ Р. ЭНДРЮС». URL: https://l.wzm.me/\_coder/custom/parallel.programming/001.htmПросмотрено: 17.11.2020

# ПРИЛОЖЕНИЕ 2 **Код программы**

// Threads.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

#include <pthread.h>

#include <string>

#include <ctime>

#include <fstream>

#include <iostream>

#include <sstream>

using namespace std;

/// <summary>

/// Структура данных для потоков

/// </summary>

struct Package {

int\* array; // Указатель на начало области обработки

unsigned int threadNum; // Номер потока

unsigned int i; //Начало последовательности

unsigned int j; //Конец последовательности

unsigned int max = 1; //Размер последовательности

unsigned int start; //Начало сектора потока

unsigned int finish; //Конец сектора

unsigned int number; //Число элементов в массиве

};

/// <summary>

/// Функция для подсчёта максимальной длины последовательности на промежутке потоком

/// </summary>

/// <param name="param">Структура данных для работы потока</param>

/// <returns>nullptr</returns>

void\* func(void\* param);

/// <summary>

/// Функция для вывода результата в файл

/// </summary>

/// <param name="threadNumber">Количество потоков</param>

/// <param name="pack">Структура данных</param>

/// <param name="answer">Путь к файлу с ответом</param>

void Write\_answer(unsigned int threadNumber, Package\* pack, const std::string& answer);

/// <summary>

/// Передача параметров в структуру данных

/// </summary>

/// <param name="pack">Структура данных</param>

/// <param name="q">Номер потока</param>

/// <param name="num">Число элементов в массиве</param>

/// <param name="threadNumber">Число птоков</param>

/// <param name="array">Маччив для изучения</param>

void Pack\_parametrs(Package\* pack, unsigned int q, unsigned int num, unsigned int threadNumber, int\* array);

/// <summary>

/// Основная программа

/// </summary>

/// <param name="args">Число аргументов командной строки</param>

/// <param name="argv">Массив аргументов командной строки</param>

/// <returns>Код ошибки</returns>

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

{

const string test = argv[1]; //путь до теста

const string answer = argv[2]; //путь до ответа

const string threds = argv[3]; //число потоков

//создание потока для чтения

ifstream fin(test);

if (!fin.is\_open()) {

throw runtime\_error("IO Exception");

}

unsigned int threadNumber = stoi(threds);

int s;

unsigned int num;

fin >> num; //число элементов в массиве

int\* array = new int[num];

unsigned int w = 0;

//Чтение элементов массива и запись их в массив

while (!fin.eof() && w < num)

{

fin >> s;

array[w] = s;

w++;

}

fin.close(); //закрытие потока

pthread\_t\* thread = new pthread\_t[threadNumber];

Package\* pack = new Package[threadNumber];

//Создание потоков

for (unsigned int q = 0; q < threadNumber; q++) {

Pack\_parametrs(pack, q, num, threadNumber, array);

pthread\_create(&thread[q], nullptr, func, (void\*)&pack[q]);

}

pthread\_join(\*thread, nullptr); //ожидание завершения всех потоков

Write\_answer(threadNumber, pack, answer); //запись ответа

//очистка памяти

delete[] thread;

delete[] array;

delete[] pack;

return 0;

}

/// <summary>

/// Передача параметров в структуру данных

/// </summary>

/// <param name="pack">Структура данных</param>

/// <param name="q">Номер потока</param>

/// <param name="num">Число элементов в массиве</param>

/// <param name="threadNumber">Число птоков</param>

/// <param name="array">Маччив для изучения</param>

void Pack\_parametrs(Package\* pack, unsigned int q, unsigned int num, unsigned int threadNumber, int\* array)

{

//Если число потоков меньше числа элементов массива

if (threadNumber <= num)

{

pack[q].start = (num / threadNumber) \* (q); //начать с ячейки массива равной данному номеру

pack[q].finish = (num / threadNumber) \* (q + 1); //закончить соответственно на начале для следующего потока

}

//Данная ситуация невозможно при корректном использовании программы, так как создание большого числа потоков

//только до определённого момента увеличивает скорость работы прогриаммы

//Так как на вход подаётся больше 1000 чисел, то создавать 1001 поток не имеет смысла

else //Если число потоков больше числа элементов массива, то, наоборот, распределять относительно числа потоков

{

pack[q].start = (threadNumber / num) \* (q) > num ? num : (threadNumber / num) \* (q);

pack[q].finish = (threadNumber / num) \* (q + 1) > num ? num : (threadNumber / num) \* (q);

}

//Запись парамметров в структуру данных

pack[q].threadNum = q;

pack[q].array = array;

pack[q].number = num;

}

/// <summary>

/// Функция для подсчёта максимальной длины последовательности на промежутке потоком

/// </summary>

/// <param name="param">Структура данных для работы потока</param>

/// <returns>nullptr</returns>

void\* func(void\* param)

{

//Восстанавливаем структуру

Package\* p = (Package\*)param;

//Начинаем идти с start

unsigned int i = p->start;

//Записываем первый для нас элемент массива

int c = p->array[i];

//Будем считать, что один элемент является последовательностью

p->i = i;

p->j=i;

unsigned int max = 0;

unsigned int i2 = 0;

//Идём до конца нашего промежутка

//Или пока наша последовательность не перестанет возрастать

while (i <= (p->finish) || ((i + 1 < p->number) && (p->array[i + 1] > c)))

{

//"Обнуляем" значения

max = 1;

i2 = i;

c = p->array[i];

//Идём пока наша последовательность возрастает

while ((i + 1 < p->number) && (p->array[i + 1] > c))

{

max++;

i++;

c = p->array[i];

}

//Записываем результат

if (max > p->max)

{

p->i = i2 == p->number ? i2 - 1 : i2;

p->j = i == p->number ? i - 1 : i;

p->max = max;

}

i++;

}

return nullptr;

}

/// <summary>

/// Функция для вывода результата в файл

/// </summary>

/// <param name="threadNumber">Количество потоков</param>

/// <param name="pack">Структура данных</param>

/// <param name="answer">Путь к файлу с ответом</param>

void Write\_answer(unsigned int threadNumber, Package\* pack, const std::string& answer)

{

unsigned int i = 0, j = 0, max = 0;

//Проходимся в цикле по результатам потоков

for (unsigned int q = 0; q < threadNumber; q++)

{

if (pack[q].max > max)

{

max = pack[q].max;

i = pack[q].i;

j = pack[q].j;

}

}

//Записываем наилучший результат в файл

fstream out(answer, ios::out);

out << "i = ";

out << i;

out << "\n";

out << "j = ";

out << j;

out << "\n";

out << "length : ";

out << max;

out.close();

}

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |