НИУ «ВШЭ»

Отчёт

Домашнее задание №3

Вариант 12

Выполнил Кан Алексей
студент группы БПИ195

Москва, 2020

Задание

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.

О программе

Работа выполнялась на языке C++. Для разработки использовалась среда разработки CLion для MacOS.

Входные данные

Пользователь поочередно вводит размер массива и количество потоков.
Предусмотрен повторный ввод при неверных входных данных.
Массив генерируется из случайных чисел.

Выходные данные

Результат работы программы выводиться пользователю в консоль.
Выводиться размер искомой последовательности и сама последовательность.

Алгоритм

Массив с входными данными делиться на части, и каждая часть обрабатывается в отдельном потоке.

После завершения работы потока мы получаем пару результатов: индекс максимального элемента последовательности и длина последовательности.

Так как, необходима наибольшая последовательность мы записываем результаты всех потоков в вектор и с помощью цикла находим наибольшую последовательность.

После этого находим индекс наименьшего элемента последовательности и выводим конечный результат.

Комментарии

В коде программы также присутствуют комментарии, поясняющие все действия.

Код программы

```
#include <iostream>
#include <vector>
#include <thread>
#include <ctime>
#include <utility>

using namespace std;

// Возваращет размер массива, введенный пользователем с проверкой на неверные данные.
int getSizeOfArray() {
    // Цикл продолжается до тех пор, пока пользователь не введет корректное значение.
    while (true)
    {
        cout << "Введите размер массива > 1000: ";
        int a;
        cin >> a;

        // Проверка на предыдущее извлечение
        if (cin.fail() || a <= 1000) // если предыдущее извлечение оказалось неудачным,
        {
            cin.clear(); // то возвращаем cin в 'обычный' режим работы
            cin.ignore(32767,'\n'); // и удаляем значения предыдущего ввода из входного буфера
            cout << "Oops, that input is invalid.  Please try again.\n";
        }
        else
        {
            cin.ignore(32767,'\n'); // удаляем лишние значения

            return a;
        }
    }
}

// Возваращет размер массива, введенный пользователем с проверкой на неверные данные.
int getCountOfThreads(int len) {
    // Цикл продолжается до тех пор, пока пользователь не введет корректное значение.
    while (true)
    {
        cout << "Введите количество потоков: ";
        int a;
        cin >> a;

        // Проверка на предыдущее извлечение
        if (cin.fail() || a >= (len / 2) || a <= 0) // если предыдущее извлечение оказалось неудачным,
        {
            cin.clear(); // то возвращаем cin в 'обычный' режим работы
```

```
            cin.ignore(32767,'\n'); // и удаляем значения предыдущего ввода
из входного буфера
            cout << "Oops, that input is invalid.  Please try again.\n";
        }
        else
        {
            cin.ignore(32767,'\n'); // удаляем лишние значения

            return a;
        }
    }
}

// Заполняет массив случайными целочисленными.
void getRandomArray(vector<int> &vec, int length) {
    srand(time(nullptr));
    for (int i = 0; i < length; ++i) {
        vec.at(i) = rand();
        cout << vec.at(i) << ", ";
    }
    cout << "\n";
}

// Поиск нужной последовательности.
pair<int, int> searchSequence(vector<int> array, int maxIndex, int minIndex)
{
    cout << "Поток " << this_thread::get_id() << " начал работу." << "\n";

    int buffer = 1, maxBuffer = 0, max = 0, maxElement = 0;
    for (int i = minIndex; i < maxIndex; i++)
    {
        if (array.at(i+1) > array.at(i))
        {
            buffer++;
            max = buffer;
            if(maxBuffer < max)
            {
                maxBuffer = max;
                maxElement = i + 1;
            }
        }
        if (array.at(i+1) <= array.at(i)) {
            max = buffer;
            if (maxBuffer < max) {
                maxBuffer = max;
                maxElement = i + 1;
            }
            buffer = 1;
        }
    }

    cout << "Поток " << this_thread::get_id() << " завершил работу." << "\n";

    return pair<int, int>(maxElement, maxBuffer);
}

int main() {
    // Создаем массив чисел.
    int len = getSizeOfArray();
    vector<int> vec(len, 0);

    getRandomArray(vec, len);

    cout << "Сгенерирован массив типа int размером " << len << ", заполненный
```

```
случайными числами.\n";

    // Получаем кол-во потоков.
    int count = getCountOfThreads(len);
    thread threads[count];

    // Делим массив на части. Каждый поток будет работать с отдельной частью.
    int temp = len / count;
    vector<int> bounds(count, 0);

    int maxIndex = temp;
    for (int i = 0; i < count; ++i) {
        bounds.at(i) = maxIndex;
        maxIndex += temp;
    }

    // Вектор результатов выполнения функций в потоках.
    // (макс. индекс последовательности, длина последовательности)
    vector< pair<int, int> > results(count, pair<int, int>(0,0));

    // Поиск нужной последовательности многопоточонстью.
    for (int i = 0; i < count; ++i) {
        threads[i] = thread(
                [&results, vec, i, temp, bounds]()
                { results.at(i) = searchSequence(vec, bounds.at(i) - 1,
bounds.at(i) - temp); }
                );
    }

    // Дожидаемся всех потоков.
    for (int i = 0; i < count; ++i) {
        threads[i].join();
    }

    int maxBuffer = 0;
    int maxElement = 0;

    // Ищем наилучший результат после работы всех потоков.
    for (int i = 0; i < count; ++i) {
        if (results.at(i).second > maxBuffer) {
            maxBuffer = results.at(i).second;
            maxElement = results.at(i).first;
        }
    }

    // Вывод длины последовательности.
    cout << "\nМаксимальная последовательность: " << maxBuffer << endl <<
endl;

    // Высчитывается минимальный элемент последовательности.
    int minElement = (maxElement + 1) - maxBuffer;

    //  Вывод последоавтельности.
    cout << "Саммая длинная последовательности чисел: ";
    for (int i = minElement; i <= maxElement; i++)
    {
        cout << "[" << i << "]" << vec.at(i) << " < ";
    }

    // Завершение программы.
    return 0;
}
```