НИУ “ВШЭ”

ОТЧЕТ

Домашнее задание 4

Вариант 12

Выполнил студент

группы БПИ195

Кан Алексей

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 Visual Studio 2019.

Входные данные

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

Выходные данные

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

Алгоритм

При помощи OpenMP алгоритм работает в нескольких потоках.

omp_set_num_threads(кол-во потоков).

Искомую последовательность находим при помощи цикла и условных выражений if.

В цикле проходимся от первого элемента до предпоследнего во входном массиве. С первого элемента начинаем новую последовательность. Если следующий элемент больше текущего, то продолжаем текущую последовательность. Если меньше или равен, то со следующего элемента начинаем новую последовательность.

С помощью if выделяем наибольшую последовательность.

Комментарии

В коде программы также присутствуют комментарии, поясняющие все действия.

Код программы

```
#include <iostream>
#include <vector>
#include <thread>
#include <ctime>
#include <utility>
#include <omp.h>

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;
        }
    }
}

// Возвараща размер массива, введенный пользователем с проверкой на неверные данные.
int getCountOfThreads() {
    // Цикл продолжается до тех пор, пока пользователь не введет корректное значение.
    while (true)
    {
        cout << "Введите количество потоков (max 16): ";
        int a;
        cin >> a;

        // Проверка на предыдущее извлечение
        if (cin.fail() || a >= 17 || 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) {
#pragma omp parallel
    {
        srand(time(nullptr));
#pragma omp for
        for (int i = 0; i < length; ++i) {
#pragma omp critical
            vec.at(i) = rand();
        }
    }
}

// Поиск нужной последовательности.
void searchSequence(vector<int> array, int *maxElement, int *maxBuffer)
{
#pragma omp parallel
    {
        //cout << "Поток " << this_thread::get_id() << " начал работу."
<< "\n";

        int buffer = 1, max = 0;
#pragma omp for
        for (int i = 0; i < array.size() - 1; i++)
        {
#pragma omp critical
            // Если следующий элемент больше, то увеличиваем длину
            // последовательности и меняем максимальный элемент.
            if (array.at(i + 1) > array.at(i))
            {
                buffer++;
                max = buffer;
                if (*maxBuffer < max) {
                    *maxBuffer = max;
                    *maxElement = i + 1;
```

```
                }
            }
            // Если следующий элемент меньше или равен, то начинаем
            // новую последовательонсть.
            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";
    }
}

int main() {
    // Поддержка кириллицы.
    setlocale(LC_ALL, "Russian");

    // Получаем кол-во потоков от пользователя.
    int count = getCountOfThreads();

    // Устанавливаем количество потоков.
    omp_set_num_threads(count);

    // Создаем вектор чисел.
    int len = getSizeOfArray();
    vector<int> vec(len, 0);
```

```
    // Заполняем его случайными числами.
    getRandomArray(vec, len);

    cout << "Сгенерирован массив типа int размером " << len << ",
заполненный случайными числами.\n";

    // Длина искомой последовательности.
    int maxBuffer;
    // Индекс максимального элемента последовательности.
    int maxElement;

    //
    searchSequence(vec, &maxElement, &maxBuffer);

    // Вывод длины последовательности.
    cout << "\nДлина максимальной последовательности: " << 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) << " < ";
    }

    cout << vec.at(maxElement) << "\n\n";

    // Пауза, чтобы программа сразу не закрывалась.
    system("pause");

    // Завершение программы.
```

```
    return 0;
}
```