ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Домашнее задание

Вариант 13

Пояснительная записка

Исполнитель
студент группы БПИ 199
М.А. Кузнецов

Москва 2020

СОДЕРЖАНИЕ

1. ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

1.1 Постановка задачи

Программа должна определить множество индексов i, для которых (A[i] - B[i]) или (A[i] + B[i]) являются простыми числами. Входные данные: массивы целых положительных чисел A и B, произвольной длины ≥ 1000. Количество потоков является входным параметром.

2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

2.1 Описание используемой модели вычислений

В качестве модели вычисления был выбран итеративный параллелизм. Его принцип заключается в работе с одним набором данных, который необходимо обработать и вычислить результат [2]. При этом действия по обработке данных одинаковы для каждой итерации и не зависят друг от друга. В последовательной программе обработка происходит поочередно, в то время как в данной модели потоки занимаются обработкой своих данных одновременно и независимо друг от друга. Они совместно работают над решением одной задачи по обработке всех данных [1]

2.2 Обоснование используемой модели вычислений

Для решения задачи был выбран именно итеративный параллелизм, так как основной задачей является обработка всех ячеек массивов A и B. Вычисление результата происходит независимо друг от друга и выполняются одни и те же действия для обработки данных, поэтому потоки занимаются обработкой сразу нескольких ячеек одновременно независимо друг от друга.

2.3 Описание алгоритма

Цикл прохода по массивам и проверки индексов разделяется между потоками по итерациям. В начале каждой итерации поток сохраняет текущий индекс в его локальную приватную переменную. После этого поток проверяет выполнения условий для сохраненного индекса. Если хотя бы одно из условий выполнилось, то выводится номер потока, который производил проверку, данный индекс и значение, которое оказалось простым.

2.4 Описание входных данных

На вход программы подается два числа в качестве аргументов командной строки:

./a.out <верхняя граница длины массивов> <число потоков>

Верхняя граница длины массива – это максимально допустимое число при генерации длины массива. Данное число должно быть не меньше 1001, так как нижняя граница равна 1000.

Число потоков не должно быть отрицательным и равным 0.

2.5 Описание выходных данных

В ходе выполнения программа выводит все элементы сгенерированных массивов.

В качестве результата программа выводит список всех индексов, которые удовлетворяют условию

3. ТЕСТЫ

3.1 Проверка верхней границы длины массива

```
rokymiel@MacBook-Pro-Mihail-2 threads_and_arrays % ./a.out 10 10
Wrong upper bound
```

Рисунок 1– Ввод малых значений

```
rokymiel@MacBook-Pro-Mihail-2 threads_and_arrays % ./a.out 1000 10
Wrong upper bound
```

Рисунок 2– Ввод 1000

3.2 Проверка числа потоков

```
rokymiel@MacBook-Pro-Mihail-2 threads_and_arrays % ./a.out 1001 0
Wrong thread number
```

Рисунок 3– Ввод 0

```
rokymiel@MacBook-Pro-Mihail-2 threads_and_arrays % ./a.out 1001 -10
Wrong thread number
```

Рисунок 4– Ввод отрицательного значения

3.3 Ввод корректных данных

Из-за большого объема выходных данных результаты тестов находятся в папке «Тесты»

4. ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

1) Практические приемы построения многопоточных приложений [Электронный ресурс] Режим доступа: http://softcraft.ru/edu/comparch/tasks/t03/ , свободный. (дата обращения: 14.11.20).
2) Итеративный параллелизм: умножение матриц [Электронный ресурс] Режим доступа: http://www.soft.architecturenet.ru/70/index-iterativnyj-parallelizm-umnozhenie-matric.htm , свободный. (дата обращения: 14.11.20).
3) Руководство по языку C++ [Электронный ресурс] Режим доступа: https://en.cppreference.com , свободный. (дата обращения: 14.11.20).
4) Многопоточное программирование. OpenMP / Примеры программ [Электронный ресурс] Режим доступа: http://softcraft.ru/edu/comparch/practice/thread/03-openmp/ , свободный. (дата обращения: 28.11.20).

ПРИЛОЖЕНИЕ

КОД ПРОГРАММЫ

```
#include <iostream>
#include <ctime>
#include "cmath"
#include <string>
#include <omp.h>

// Кузнецов Михаил Алексндрович БПИ199
// 13 вариант

uint32_t* a; // Массив A
uint32_t* b; // Массив B
uint32_t arrays_size; // Хранит размер массивов
uint32_t thread_num = 6; // Число потоков
uint32_t upperBound; // Верхняя граница длины массива

/// Генерирует длину массива
void generateSize() {
    arrays_size = rand() % (upperBound - 999) + 1000;
}

/// Генерирует массив
/// \return Сгенерированный массив
uint32_t* generateArray() {
    uint32_t* array = new uint32_t[arrays_size];
    for (int j = 0; j < arrays_size; ++j) {
        array[j] = (unsigned)rand();
    }
    return array;
}

/// Проверяет является ли переданное число простым
/// \param num Число для проверки на простоту
/// \return Результат проверки
bool isPrime(long num, int* value) {
    if (num == 0 || abs(num) == 1) return false;
    for (long j = 2; j <= sqrt(abs(num)); ++j) {
        if (num % j == 0) return false;
    }
    *value = num;
    return true;
}

/// Выводит массив
/// \param arrayName Название массива
/// \param ar Массив
void printArray(std::string arrayName, uint32_t* ar) {
    for (int j = 0; j < arrays_size; ++j) {
        std::cout << arrayName << "[" << j << "] = " << ar[j] << std::endl;
    }
```

```
}


int main(int argc, char** argv) {
    srand((unsigned)time(NULL));
    if (argc != 3) { // Проверяем число аргументов
        std::cout << "Wrong number of console arguments" << std::endl;
        return 1;
    }
    if (std::stol(argv[1]) < 1001) { // Проверяем верхнюю границу
        std::cout << "Wrong upper bound" << std::endl;
        return 1;
    }
    if (std::stol(argv[2]) <= 0) { // Проверяем число потоков
        std::cout << "Wrong thread number" << std::endl;
        return 1;
    }
    upperBound = std::stoul(argv[1]); // Получаем верхнюю границу
    thread_num = std::stoul(argv[2]); // Получаем число потоков

    generateSize();
    a = generateArray();
    b = generateArray();
    printArray("a", a);
    printArray("b", b);
    std::cout << "Set of indices i:" << std::endl;

    // Ищем индексы массивов подходящие под условия
#pragma omp parallel num_threads(thread_num)
    {
        auto threadNum = omp_get_thread_num();
        int c;
        int primeValue;
#pragma omp for private(c,primeValue)
        for (int j = 0;j < arrays_size; j++)
        {
            c = j;
            if (isPrime(a[c] + b[c], &primeValue) || isPrime(a[c] - b[c], &primeValue)) {
#pragma omp  critical
                {
                    std::cout << threadNum << " thread " << "i: " << c << " with prime
result: " << primeValue << std::endl;
                }
            }
        }
    }
    delete[] a;
    delete[] b;
    return 0;
}
```