**ЗАДАНИЕ 26.11.2021**.  
Поиск простых чисел в диапазоне [a, b] с использованием std::async и std::future

**Выполнил:**

студент 3 курса 13 группы кафедры ТП.

Петров Андрей Александрович

Реализуем последовательную программу для поиска простых чисел в диапазоне [a, b].

#include **<iostream>**#include **<vector>  
  
using namespace** std;  
  
**const int** a = 1;  
**const int** b = 10000;  
vector<**int**> primeNumbers;  
  
**bool** isPrime(**int** n) {  
 **if** (n <= 1) **return false**;  
  
 **for** (**int** i = 2; i < n; i++) {  
 **if** (n % i == 0) **return false**;  
 }  
 **return true**;  
}  
  
**int** main() {  
 **double** start\_time = clock();  
 **for** (**int** i = a; i <= b; i++) {  
 **if** (isPrime(i)) {  
 primeNumbers.push\_back(i);  
 }  
 }  
 **double** end\_time = clock();  
 **double** exec\_time = (end\_time - start\_time) / **CLOCKS\_PER\_SEC**;  
 cout << **"Total time: "** << exec\_time << **"s"** << **"\n"**;  
 printf(**"Prime numbers on the [%d; %d]: %d"**, a, b, primeNumbers.size());  
}

Реализуем параллельную программу для поиска простых чисел в диапазоне [a, b] с использованием std::async и std::future.

#include **<iostream>**#include **<future>**#include **<vector>  
  
using namespace** std;  
  
**const int** a = 1;  
**const int** b = 10000;  
**const int** thread\_number = 2;  
  
vector<future<vector<**int**>>> futures;  
vector<**int**> futurePrimeNumbers;  
  
**bool** isPrime(**int** n) {  
 **if** (n <= 1) **return false**;  
  
 **for** (**int** i = 2; i < n; i++) {  
 **if** (n % i == 0) **return false**;  
 }  
 **return true**;  
}  
  
vector<**int**> findNumberPrimes(**int** start, **int** end) {  
 vector<**int**> primeNumbers;  
 **for** (**int** i = start; i <= end; i++) {  
 **if** (isPrime(i)) {  
 primeNumbers.push\_back(i);  
 }  
 }  
 **return** primeNumbers;  
}  
  
**int** main() {  
 **double** start\_time = clock();  
 **int** step = (b - a) / thread\_number;  
  
 **for** (**int** i = 0; i < thread\_number; i++) {  
 futures.push\_back(async(launch::***async***, findNumberPrimes, step \* i +

a, step \* (i + 1) + a - 1));  
 }  
  
 **for** (**auto** &future: futures) {  
 **auto** res = future.get();  
 futurePrimeNumbers.insert(futurePrimeNumbers.end(),

res.begin(), res.end());  
 }  
 **double** end\_time = clock();  
 **double** exec\_time = (end\_time - start\_time) / **CLOCKS\_PER\_SEC**;  
 cout << **"Total time: "** << exec\_time << **"s"** << **"\n"**;  
 printf(**"Prime numbers on the [%d; %d]: %d"**, a, b,

futurePrimeNumbers.size());  
}

Произведем вычислительные эксперименты.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Диапазон, [a,b]** | **Последовательная программа, с.** | **Параллельная программа, с.** | | | | | |
| **2 потока** | | | **4 потока** | | |
| **Время, с.** | **Ускорение** | **Эффективность** | **Время, с.** | **Ускорение** | **Эффективность** |
| [1, 10000] | 0.023 | 0.016 | 1.4375 | 0.71875 | 0.011 | 2.0909 | 0.52273 |
| [1, 100000] | 1.348 | 1.036 | 1.3012 | 0.65058 | 0.744 | 1.8118 | 0.45296 |
| [1, 1000000] | 109.214 | 85.132 | 1.2829 | 0.64144 | 58.682 | 1.8611 | 0.46528 |