Przetwarzanie współbieżne

Laboratorium 4

Jan Bartula

# Zadanie 1

## Opis

Do obliczania liczb pierwszych wystarczy liczyć wielokrotności:

od 2 do pierwiastka z N (włącznie)

Do implementacji wielowątkowości, wydzieliłem funkcję zaznaczającą wielokrotności liczby i ją uruchamiam w osobnych wątkach:

void mark\_multiples(long number) {  
 long temp = number \* number;  
 while (temp <= max\_number) {  
 is\_not\_prime[temp] = true;  
 temp = temp + number;  
 }  
}

Zamiast przechowywania tablicy liczb, przechowuję wektor zmiennych typu **bool**, gdzie Falseoznacza – że liczba jest wielokrotnością (więc nie jest pierwszą), pozostałe są więc liczbami pierwszymi.

Teoretycznie można by zastosować tutaj mutexy, ale jest to zbędne, ponieważ nawet gdyby dwa wątki w jednym czasie próbowały ustawić, że pod danym indeksem (odpowiadającym liczbie) nie może istnieć liczba pierwsza, to i tak ustawią one na tą samą wartość. Więc nie istnieje tutaj t.zw. sekcja krytyczna.

Program ten mógłby też zrzucać liczby pierwsze do wektora, jednak wtedy istniałaby już sekcja krytyczna związana z wstawianiem liczb pierwszych do wektora. Obrana metoda wydaje się wydajna i optymalna.

Program na początku odczytuje z parametrów liczbę, do której mamy wyznaczać liczby pierwsze

if (argc != 2) {

cout << "Run with: <max\_number:int>";

max\_number = 100;

} else {

max\_number = strtol(argv[1], nullptr, 10);

}

int square\_root = (int) round(sqrt(max\_number));

Następnie tworzy wektor wątków I tablicę znaczników wielokrotności

is\_not\_prime = new bool[max\_number];  
vector<thread> threads(square\_root - 1);  
  
cout << "Spawn thread multiples of: " << endl << "\t";  
for (int i = 2; i <= square\_root; i++) {  
 cout << i << " ";  
 threads[i - 2] = thread(mark\_multiples, i);  
}  
cout << endl;

Następnie czekamy na zakończenie pracy wątków

for (auto &thread: threads) {

thread.join();

}

I wypisujemy liczby pierwsze

int count = 0;  
for (int number = 2; number < max\_number; number++) {  
 if (!is\_not\_prime[number]) {  
 cout << setw(4) << number << ",";  
 count++;  
 if (count % 16 == 0) cout << endl << "\t";  
 }  
}

## Przykłady

### ./lab5/zad1 100

Marking multiples of numbers up to:

10

Spawn thread multiples of:

2 3 4 5 6 7 8 9 10

Prime numbers:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,

59, 61, 67, 71, 73, 79, 83, 89, 97,

Count of primes:

25

### ./lab5/zad1 10000

Count of primes:

1229

(sprawdzone z wynikami dostępnymi w internecie)

### ./lab5/zad1 1009

947, 953, 967, 971, 977, 983, 991, 997,1009,

Count of primes:

169

Przykład krańcowy 1009 jest zawarty I liczba liczb pierwszych się zgadza.