Zajęcia IX

Cele:

zapoznać się z:

- wielowątkowością

Problem

Wszystkie dotychczasowe programy korzystały tylko z jednego rdzenia procesora, czyli `12.5%` z `8-rdzeniowego` procesora.

Błyskotliwe rozwiązanie

Skorzystać z reszty rdzeni!

```cpp
#include <thread>
#include <iostream>
#include <vector>

#define THREADS 8

void thread_function(int id, int x) {
        std::cout << "Hello from thread " << id << "!" << std::endl;
}

int main() {
        std::vector<std::thread> threads;

        for (int i = 0; i < THREADS; ++i)
        {
                std::thread t(thread_function, i, 2);
                threads.push_back(std::move(t));
        }

        std::cout << "Hello from the main thread!" << std::endl;

        for (auto &thread : threads)
                thread.join();

        return 0;
}
```

Problem wielowątkowości

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

#define THREADS 8
int gloabl_variable = 0;

void thread_function(int id) {
        std::cout << "Hello from thread " << id << "!" << std::endl;
        for (int i = 0; i < 1000000; i++)
                gloabl_variable += id;
}

int main() {
        std::vector<std::thread> threads;

        for (int i = 0; i < THREADS; ++i)
        {
                std::thread t(thread_function, i);
                threads.push_back(std::move(t));
        }

        for (auto &thread : threads)
                thread.join();

        std::cout << "Hello from the main thread! Global variable:" << gloabl_variable << std::endl;

        return 0;
}
```

Zmienna `gloabl_variable` wydrukowana z wątku głównego powinna mieć wartość:

`1000000*(0+1+2+3+4+5+6+7) = 28 milionów`

A ma znacznie mniej i nie zawsze tyle samo.

Zjawisko **race condition**

Rozwiązanie dla race condition

`std::mutex` 🎉✨

```cpp
#include <mutex>

std::mutex mtx;

void thread_function(int id)
{
        std::cout << "Hello from thread " << id << "!" << std::endl;
        for (int i = 0; i < 1000000; i++)
        {
                mtx.lock();
                gloabl_variable += id;
                mtx.unlock();
        }
}
```

Zadanie Lab I

Napisz program, który:

- wczyta z pliku wiele linii z ciągami DNA, każdy ciąg w osobnej linii
- użyje `std::thread` do policzenia ilości poszczególnych nukleotydów w danym ciągu DNA
- wynik zapisze do pliku

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <thread>
#include <map>

void count_nucleotides(const std::string dna, std::map<char, int> &result)
{
	std::map<char, int> local_count = {{'A', 0}, {'T', 0}, {'C', 0}, {'G', 0}};
	for (char nucleotide : dna)
		local_count[nucleotide]++;

	result = local_count;
}

int main()
{
	std::ifstream infile("dna_input.txt");
	std::ofstream outfile("dna_output.txt");
	std::vector<std::string> dna_lines;
	std::string line;

	while (std::getline(infile, line))
		dna_lines.push_back(line);

	std::vector<std::thread> threads;
	std::vector<std::map<char, int>> results(dna_lines.size());

	for (size_t i = 0; i < dna_lines.size(); i++)
		threads.emplace_back(count_nucleotides, dna_lines[i], std::ref(results[i]));

	for (std::thread &t : threads)
		t.join();

	for (size_t i = 0; i < results.size(); ++i)
	{
		outfile << "Line! " << i + 1 << ": ";
		for (std::pair<const char, int> &pair : results[i])
			outfile << pair.first << "=" << pair.second << " ";
		outfile << "\n";
	}

	return 0;
}
```

Zadanie I

Napisz program, który:

- wczyta z pliku jedną linię *bardzo* długiego ciągu DNA
- użyje `std::thread` do policzenia ilości poszczególnych nukleotydów w danym ciągu DNA
- wynik wydrukuje na wyjście standardowe
- program musi wykonywać się szybciej niż podany niżej program
  - (do pomiaru czasu wykonywania, na linuxie możesz użyć komendy `time`)

```cpp
#include <iostream>
#include <fstream>

int main() {
	std::ifstream infile("long_dna.txt", std::ios::in);
	if (!infile)
		return 1;

	int countA = 0, countC = 0, countG = 0, countT = 0;
	char c;

	while (infile.get(c)) {
		switch (c)
		{
		case 'A':
			countA++;
			break;
		case 'C':
			countC++;
			break;
		case 'G':
			countG++;
			break;
		case 'T':
			countT++;
			break;
		}
	}
	std::cout << "A: " << countA << "\n";
	std::cout << "C: " << countC << "\n";
	std::cout << "G: " << countG << "\n";
	std::cout << "T: " << countT << "\n";
	return 0;
}
```