Zajęcia X

Cele:

zapoznać się z:

- szablonami

Problem

Chcemy policzć zawartość `GC` w ciągu. Do tego tworzymy oczywiście funkcję. Ale co jeśli czasem nasze dna jest przechowywane w różnych typach zmiennych? (np. `std::string`, `std::vector<char>`, albo we własnej klasie)

Szablony

Umożliwiają pisanie uniwersalnych funkcji i klas działających z różnymi typami danych.

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

template<typename T>
double gc_content(const T& seq) {
    int gc_count = 0;
    for (char base : seq) {
        if (base == 'G' || base == 'C') {
            ++gc_count;
        }
    }
    return (seq.empty() ? 0.0 : 100.0 * gc_count / seq.size());
}

int main() {
    std::string dna = "AGCTGGGCCCAA";
    std::vector<char> rna = {'A', 'G', 'C', 'U', 'G', 'G', 'C'};

    std::cout << "GC-content DNA: " << gc_content(dna) << std::endl;
    std::cout << "GC-content RNA: " << gc_content(rna) << std::endl;

    return 0;
}
```

Zadanie Lab I

Napisz program, który korzystając z szablonów zaimplementuje `std::array`

```cpp
#include <iostream>

template <typename T, int N>
class MyArray
{
	T data[N];

public:
	T &operator[](int index)
	{
		return data[index];
	}

	int size() const
	{
		return N;
	}

	T *begin()
	{
		return data;
	}

	T *end()
	{
		return data + N;
	}
};

int main()
{
	MyArray<int, 5> arr;

	for (int i = 0; i < arr.size(); ++i)
		arr[i] = i * 2;

	for (auto x : arr)
		std::cout << x << " ";

	return 0;
}
```

Zadanie I

Napisz program, który:

- korzystając z szablonów, zaimplementuje funkcję liczącą ilość danych nukleotydów (A,T,C i G), która obsłuży `std::string`, `std::vector<char>` oraz stworzoną przez siebie klasę `MiniDna` (stwórz nową, nie korzystaj z klasy z porzedniego laboratorium)
- funkcja `main` powinna wyglądać:

```cpp
int main() {
    std::string dna1 = "AGCTGGGCCCAA";
    std::vector<char> dna2 = {'A', 'G', 'C', 'U', 'G', 'G', 'C'};
    MiniDna dna3("GGGCGCGTTA");

    std::cout << "GC-content std::string: " << gc_content(dna1) << std::endl;
    std::cout << "GC-content std::vector: " << gc_content(dna2) << std::endl;
    std::cout << "GC-content MiniDna:     " << gc_content(dna3) << std::endl;

    return 0;
}
```

Zadanie II

Napisz program, który:

- wykorzystując wskaźniki (bez `std::string` ani żadnych wbudowanych funkcji) i ich arytmetykę znajdzie najdłuższy homopolimer w DNA.
- będzie zawierał funkcję o sygnaturze:

- `char* find_longest_homopolymer(char* dna, int length, int* out_len)`
  - funkcja ma zwrócić wskaźnik na pierwszy element homopolimeru
  - funkcja przyjmuje DNA jako wskaźnik na `char`
  - funkcja przyjmuje `length` ciągu DNA (opcjonalny, można pominąć ten argument)
  - pod zmienną na którą wskazuje `out_len` należy podstawić długość znalezionego podciągu

Przykład

Wejście:

`ATTCG`

Wyjście:

- wskaźnik zwrócony z funkcji wskazuje na pierwszą literę `T` o indeksie 1, oraz pod `out_len` jest wartość 2

Przykład

Wejście:

`AAATTTGGGGGCCCCAAAGG`

Wyjście:

- wskaźnik zwrócony z funkcji wskazuje na pierwszą literę `G` o indeksie 6, oraz pod `out_len` jest wartość 5