Zajęcia VII

Cele:

zapoznać się z:

- std::array
- std::vector
- std::map
- std::set

std::array

```
#include <stdio.h>
#define ARRAY_SIZE 5

int main() {
  int arr[ARRAY_SIZE] = {1,2,100,-13,0};

  printf("Array size: %d \r\n", ARRAY_SIZE);

  return 0;
}
```

std::array

```
#include <stdio.h>
#define ARRAY_SIZE 5
#define ARR_SIZE 4
#define DNA_SIZE 3

int main() {
  int arr[ARRAY_SIZE] = {1,2,100,-13,0};
  char dna[DNA_SIZE] = {'A', 'T', 'C'};
  int int_arr[ARR_SIZE] = {0,1,3};

  printf("Array size: %d \r\n", ARRAY_SIZE);
  printf("Arr size: %d \r\n", ARR_SIZE);

  return 0;
}
```

std::array

```cpp
#include <iostream>
#include <array>

int main() {
  std::array<int, 5> arr = {1,2,100,-13,0};

  std::cout << arr.size() << std::endl;
  return 0;
}
```

std::array

Co zyskujemy:

- `array.size()`
- `array.at(i) lub array[i]`
- `array.fill(x)`
- `array.front() oraz array.back()`

std::vector

```cpp
#include <iostream>
#include <fstream>
std::ifstream file("pliczek.txt");

int main() {
  std::string dnas[10];

  int i = 0;
  while(std::getline(file, dnas[i++]));

  return 0;
}
```

Plik z <10 liniami: 😊

Plik z >10 liniami: 😕

std::vector

```
#include <iostream>
#include <fstream>
#include <vector>
std::ifstream file("pliczek.txt");

int main() {
  std::vector<std::string> dnas;

  std::string line;
  while (std::getline(file, line)) {
      dnas.push_back(line);
  }
  return 0;
}
```

Zawsze: 😊

Co zyskujemy:

- `push_back(x)` - dodaje element na koniec
- `pop_back()` - usuwa ostatni element
- `size()` - zwraca liczbę elementów
- `capacity()` - zwraca liczbę elementów, które mogą się zmieścić bez realokacji
- `empty()` - sprawdza, czy wektor jest pusty
- `clear()` - usuwa wszystkie elementy
- `operator[i]` - szybki dostęp do `i` -tego elementu
- `front()` - pierwszy element
- `back()` - ostatni element

- `insert(i, value)` - wstawia element w dowolne miejsce
- `erase(i)` - usuwa element z dowolnego miejsca
- `shrink_to_fit()` - zwalnia nieużywaną pamięć

std::map

```cpp
#include <iostream>
#include <map>

int main() {
  std::map<std::string, int> dnas;

  dnas["CTG"] = 3;
  dnas["ATT"] = 0;

  std::cout << dnas["ATT"] << std::endl;
  return 0;
}
```

std::map

```cpp
#include <iostream>
#include <map>

int main() {
  std::map<std::string, char> codon = {
    {"UAG", '|'},
    {"UUU", 'F'},
    // ...
  };

  std::string rna = "ATTCGUUUUAG";
  return 0;
}
```

std::set

```cpp
#include <iostream>
#include <set>

int main() {

  std::set<char> nucleotides = {'A', 'T', 'C', 'G'};
  std::string dna = "ATTCG";

  for( char nucleotide : dna ) {
    nucleotides.insert(nucleotide);
  }

  if( nucleotides.size() > 4 )
    std::cout << "Invalid dna!" << std::endl;

  return 0;
}
```

Zadanie I

Napisz program, który:

- przyjmie ciąg DNA na wejściu oraz jedną liczbę `k`.
- wypisze wszystkie możliwe `k`-mery, każdy w osobnej linii
- musi opierać się o wskaźniki i ich arytmetykę
- cały program nie może zawierać się w `main`!

Przykład

Wejście:

```
ACGTACGT
3
```

Wyjście:

```
ACG
CGT
GTA
TAC
ACG
CGT
```

Zadanie II

Napisz program, który:

- dokona przetłumaczenia RNA na ciąg protein
- program wczyta z wejścia ciąg RNA i wyprodukuje na wyjściu ciąg protein
- skorzysta z nowych struktur (array, vector, map lub/i set)
- pamiętaj obsłużyć małe i duże litery oraz błędny, wprowadzony ciąg
- obsłuż STOP na przykład znakiem `'|'`

Przykład

Wejście:

```
UGUCCUCCAUAA
```

Wyjście:

```
CPP |
```

Zadanie III

Napisz program, który:

- wczyta z pliku ciągi DNA
- policzy, jaka jest maksymalna ilość duplikatów
- skorzysta z nowych struktur (array, vector, map lub/i set)
- ciąg `ATT` i `aTt` są duplikatami

Przykład

Plik:

```
AT
CGT
ggg
at
CGt
ATTCG
GGG
GGG
```

Wyjście:

```
3
```

Zadanie IV

Napisz program, który:

- wczyta z pliku sekwencje DNA (ciąg per linia)
- wczyta od użytkownika z wejścia motyw (podciąg)
- wypisze indeksy dla których znaleziono podany przez użytkownika motyw
- zignoruj błędne ciągi w pliku
- nie ignoruj wprowadzonego błędnego motywu
- skorzysta z nowych struktur (array, vector, map lub/i set)

Przykład

Plik:

```
CGCATC
ATTCG
AATCTGGGG
```

Wejście:

```
ATC
```

Wyjście:

```
0, 2
```