Zajęcia VIII

Cele:

zapoznać się z:

- dzieleniem projektu na pliki
- całkowitą podstawą klas:
  - deklaracja klasy
  - obiekt a klasa
  - konstruktor
  - metody

Problem

Trzeba napisać program, który będzie składał się z dziesiątek funkcji, mnóstwa zmiennych, będzie importował wiele bibliotek. I to wszystko w jednym pliku `main.cpp`? ❌

Rozwiązanie

Podzielić kod na pliki!

```cpp
#include <iostream>

double calculateGC(std::string& dna) {
  // hard calculations
  return 0.52;
}

int main() {
  std::string dna1("ATTCG");

  std::cout << calculateGC(dna1) << std::endl;

  return 0;
}
```

main.cpp

```cpp
#include "gc_calculator.h"

int main() {
  std::string dna1("ATTCG");

  std::cout << calculateGC(dna1) << std::endl;

  return 0;
}
```

gc_calculator.h

```
#pragma once

#include <iostream>

double calculateGC(std::string& dna);
```

gc_calculator.cpp

```cpp
#include "gc_calculator.h"

double calculateGC(std::string& dna) {
  // hard calculations
  return 0.52;
}
```

Kompilacja

Użycie IDE, lub ręczna kompilacja:

```
g++ -Wall -o main main.cpp gc_calculator.cpp
```

Struktury na dopalaczu

```
struct dna {
    int a;
    int t;
    int c;
    int g;

    float gc;
};  // note semicolon
```

Metoda (czyli funkcja w klasie)

Dna.h

```
class Dna {
  public:
    int a,t,c,g;

    float gc();
};    // note semicolon
```

Dna.cpp

```
#include "Dna.h"

float Dna::gc() {
    return (float)(this->g + this->c) / (this->a + this->t + this->g + this->c);
}
```

main.cpp

```cpp
#include <iostream>
#include "Dna.h"

int main() {
  Dna my_dna;

  my_dna.a = 1;
  my_dna.t = 2;
  my_dna.c = 3;
  my_dna.g = 4;

  std::cout << my_dna.gc() << std::endl;

  return 0;
}
```

Konstruktor

```
class Dna {
  public:
    int a,t,c,g;

    Dna(std::string& input_dna);
    Dna();

    float gc();
};
```

```
Dna::Dna(std::string& input_dna) {
  this->a = 3;
  std::cout << this->gc() << std::endl;
}
```

Prywatne metody i zmienne

```
class Dna {
  private:
    int a,t,c,g;

  public:
      Dna(std::string& input_dna);
      Dna();

      float gc();
};
```

```
Dna my_dna;
std::cout << my_dna.a << std::endl;
```

To jest szablon

```
class Dna {
  private:
    int a,t,c,g;

  public:
      Dna(std::string& input_dna);
      Dna();

      float gc();
};
```

A to jest obiekt

```
Dna my_dna;
```

A to są obiekty!

```
Dna my_dna;
Dna other_dna;
Dna array_of_dnas[10];
```

Zadanie I

Napisz program, w którym będzie zdefiniowana klasa do kompleksowej obsługi DNA.
Wymagania:

- program będzie podzielony na co najmniej 3 pliki - (main.cpp, dna.cpp, dna.h)
- klasa musi mieć trzy konstruktory, jeden przyjmuje `std::string&`, drugi `char *`, a trzeci plik (jako deskryptor albo ścieżkę do pliku)

- klasa musi mieć metody:
  - do obliczania zawartości GC o nazwie `gc`
  - do zwrócenia komplementarnego ciągu (odwrotnego) o nazwie `reverse_complement`
  - do zwrócenia ciągu po transkrypcji o nazwie `transcribe`
  - prywatną metodę do weryfikacji poprawności ciągu o nazwie `verify`
  - do obliczenia dystansu Hamminga o nazwie `distance`, która przyjmie referencję do klasy zadeklarowanej w `dna.h`

- funkcja `main` powinna zaprezentować możliwości klasy (użyć wszystkich możliwych metod, ładnie wydrukować ich resultaty)
- pamiętaj nie duplikować kodu!