02. Struktury w języku C++

1 Struktury

Strukturą (ang. *structure*) nazywamy zbiór danych zgrupowanych pod jedną nazwą. Pozwalają one reprezentować bardziej złożone obiekty niż liczby i znaki.

```
struct nazwa_struktury {
    typ_1 nazwa_pola_1;
    typ_2 nazwa_pola_2;
    ...
}; // wymagany srednik!
```

Załóżmy, że chcemy przechować informacje dotyczące osoby. Jeżeli będą nam potrzebne imię, nazwisko i wiek, to konieczne będzie stworzenie trzech różnych zmiennych.

Niesie to ze sobą pewne negatywne konsekwencje, np.: we wszystkich funkcjach operujących na osobach, konieczne będzie przekazywanie wszystkich atrybutów osoby za pomocą trzech argumentów. Gdyby była potrzebna lista osób, to potrzebowalibyśmy aż trzech tablic (odpowiednio dla imion, nazwisk i wieku).

Lepszym pomysłem jest utworzenie złożonego typu danych, który mógłby przechowywać te trzy informacje naraz. Umożliwiają to struktury:

```
struct Person {
    std::string fname;
    std::string sname;
    int age;
};
```

Zadanie 00 Zaprojektuj struktury, które posłużą do reprezentowania: daty, liczby zespolonej, ułamka zwykłego, użytkownika portalu społecznościowego, filmu, pliku, dokumentu HTML. Następnie dla każdej z powyższych struktur zaproponuj przydatne funkcje, które będą na nich operowały.

Dostęp do *składowych* (ang. *members*) struktury (nazywanych również *atrybutami*) uzyskuje się poprzez operator kropki:

```
Person p;

p.fname = "Foo";
p.sname = "Bar";
p.age = 12;

std::cout << p.fname << " " << p.sname << p.age;
```

Zadanie 01 Zdefiniuj strukturę `Point`, która będzie reprezentowała punkt w układzie współrzędnych i posiadała składowe `x` oraz `y` typu `float`.

Zadanie 02 Napisz funkcję `void print_point(Point p)`, która wypisze punkt np. w postaci `(1.0, 2.5)`.

2 Wskaźniki do struktur

Tak jak w przypadku wszystkich innych typów danych, możliwe jest również tworzenie wskaźników do struktur:

```
struct Movie {
    std::string title;
    int year;
};

Movie movie;
Movie *ptr_movie = &movie;
```

Dostęp do składowych struktury, która jest przechowywana jako wskaźnik możliwy jest na dwa sposoby: poprzez wyłuskanie struktury i zastosowanie operatora kropki `.` lub poprzez operator strzałki `->`:

```
(*ptr_movie).title = "Foo"; // wyluskanie i operator kropki
ptr_movie->year = 2014;     // operator strzalki na wskazniku
```

Przykładowa funkcja inicjalizująca strukturę typu `Movie` podanym tytułem i rokiem:

```
void init_movie(Movie *movie, std::string title, int year) {
  movie->title = title;
  movie->year = year;
}
```

Zadanie 03 Napisz funkcję `init_point` inicjalizującą punkt `Point` podanymi wartościami `x` i `y`. Przetestuj działanie funkcji.

Zadanie 04 Napisz funkcję `float dist(Point p1, Point p2)`, która zwróci odległość euklidesową między dwoma punktami `p1` i `p2`. Przetestuj działanie funkcji.

Zadanie 05 Stwórz dynamicznie tablicę punktów o rozmiarze i współrzędnych podawanych przez użytkownika. Pamiętaj o zwolnieniu pamięci.

Zadanie 06 Napisz funkcję, która z tablicy punktów reprezentujących linię łamaną (ścieżkę) obliczy jej długość całkowitą. Wykorzystaj wcześniej zdefiniowaną funkcję `dist`. Przetestuj działanie funkcji.

3 Zagnieżdżanie struktur

Z zagnieżdżaniem struktur mamy do czynienia w przypadku, gdy składową jednej struktury jest inna struktura, np.:

```
struct Movie {
    std::string title;
    int year;
    Person director;
} movie;
```

Przykładowa inicjalizacja struktury `movie`:

```
movie.title = "The Amaizing Spider-Man 2";
movie.year = 2014;

Person p;
p.fname = "Marc";
p.sname = "Webb";

movie.director = p;
```

Zadanie domowe 07 (1 pkt) Zdefiniuj strukturę `Path` reprezentującą ścieżkę jako tablicę punktów `Point`. Tablica powinna być dynamicznie inicjalizowana przy pomocy funkcji `void init_path (Path* p, int size)`, gdzie `size` określa maksymalną liczbę punktów. Za zwolnienie zajętej pamięci powinna odpowiadać funkcja o nazwie `void del_path(Path* p)`. Następnie zdefiniuj funkcję, która obliczy całkowitą długość ścieżki (patrz poprzednie zadanie).

Napisz fragment kodu demonstrujący działanie powyższej struktury i wszystkich funkcji.