Laboratorium 2 – Konstruktory

Zad. 1

W klasie Pies zdefiniuj pusty konstruktor, który na razie nic nie będzie robił i nie będzie pobierał żadnych parametrów.

- Zadeklaruj obiekt typu Pies (np. wilczur):
  ```
  Pies wilczur;
  ```
- Używając konstruktora Pies() stwórz obiekt typu Pies (np. burek):
  ```
  Pies burek = Pies();
  ```
  (jest równoważne używanej wcześniej definicji `Pies burek;`)
- Używając operatora new i konstruktora Pies() stwórz wskaźnik wsk na obiekt typu Pies:
  ```
  Pies *wsk = new Pies();
  ```

Obiektowi burek oraz obiektowi wskazywanemu przez wsk nadaj imię i przypisz barwę, a następnie wykonaj dla nich metody Spaceruj() i Jedz(). Wskazówka: w pierwszym przypadku dostęp do zmiennych i metod przez kropkę `.`, a w drugim przez strzałkę `->`.

Zad. 2

Sprawdź, jakie błędy zostaną zasygnalizowane przy kompilacji projektu:

- gdy przeniesiesz konstruktor z sekcji public do sekcji private,
- gdy zakomentujesz definicję konstruktora w klasie Pies.

Zad.3

Do klasy Pies dopisz pole energia (typ liczbowy). W definicji bezparametrowego **konstruktora domyślnego** dopisz polecenie, które podczas wywołania konstruktora przypisze zmiennej energia jakąś **początkową wartość domyślną** (np. 50). Następnie sprawdź, ile wynosi wartość energii dla wszystkich dotychczas stworzonych obiektów typu Pies (tzn. dla obiektów wilczur, burek i obiektu wskazywanego przez wsk).

Zad. 4

Zdefiniuj drugi konstruktor z parametrem wejściowym energia_poczatkowa, który będzie inicjalizował wartość zmiennej energia danego obiektu. Stwórz czwartego psa za pomocą tego konstruktora, podając wartość parametru energia_początkowa, np.

```
Pies husky = Pies(70);
lub Pies husky(70);
```

Zauważ, że w jednym programie możesz użyć obu napisanych dotychczas konstruktorów (czyli z jednym parametrem i bezparametrowego). Taką operację nazywa się **przeciążaniem (przeładowaniem)** konstruktora.

Zad. 5

Zmodyfikuj metody Spaceruj() i Jedz(), aby pierwsza obniżała energię danego psa, a druga ją podwyższała. Sygnalizuj aktualny poziom energii psa po wykonaniu każdej z metod (umieść komunikat wewnątrz każdej z funkcji lub stwórz dodatkową metodę IleEnergii() zwracającą wartość zmiennej energia).

Zad. 6

Zdefiniuj trzeci konstruktor z parametrami wejściowymi energia_poczatkowa i imię. Użyj tego konstruktora w programie do zainicjowania nowego obiektu, np.

```
Pies husky = Pies(70,”Max”);
lub Pies husky(70,”Max”);
```

Zad. 7

Zdefiniuj czwarty konstruktor z parametrami wejściowymi energia_poczatkowa i barwa. Użyj tego konstruktora w programie do zainicjowania nowego obiektu, np.

```
Pies jamnik = Pies(bialy, 20);
lub Pies jamnik(bialy, 20);
```

Zad. 8

Dla obiektu husky (który oprócz energii i imienia ma przypisaną barwę) wykorzystaj **niejawny konstruktor kopiujący** i stwórz jego kopię, np.:

```
Pies wilk(husky);
```

Następnie zmień wilkowi imię lub barwę i sprawdź, czy zmiana ta miałą wpływ na obiekt husky.

Zad. 9

Zmodyfikuj **konstruktor kopiujący:**

```
Pies(const Pies &pies)
            {
                 imie = pies.imie;
                 barwa = pies.barwa;
                energia = pies.energia;
            }
```

aby kopia obiektu (np. wilk) miała większą energię (np. +3) od oryginału. Wywołaj konstruktor kopiujący:

```
Pies wilk(husky);
```

A następnie sprawdź, jaką energię ma wilk i husky.

Zad. 10*

Zmodyfikuj konstruktor kopiujący z zadania 9 tak, aby zmienna imię kopii obiektu (np. drugi_husky) była zlepkiem imienia obiektu oryginalnego (np. husky.imie to ”Max”) i ciągu znaków (np. „_2”), czyli po wywołaniu konstruktora

```
Pies drugi_husky(husky);
```

pod drugi_husky.imie ma kryć się „Max_2”. Takie zlepianie ciągów znaków jest nazywane konkatenacją.
Wskazówka: można użyć funkcji `strcpy` i `strcat` z biblioteki string.h oraz operatora new.

Pytania:

- Jaką nazwę musi mieć konstruktor?
- Czy konstruktor może się znajdować w sekcji private?
- Ile konstruktorów może mieć jedna klasa?
- Czy klasa może mieć dwa konstruktory o takiej samej liczbie parametrów?
- Czy możemy skorzystać z konstruktora kopiującego, jeżeli nie został on jawnie zdefiniowany dla danej klasy?