Laboratorium 3 – klasy

Klasa to typ zdefiniowany przez użytkownika zbudowany z dwóch podstawowych rodzajów składowych: pól („zmiennych wewnątrz klasy”) i metod („funkcji wewnątrz klasy”). Poniżej przedstawiono przykładową deklarację klasy:

```
class Klasa {
    int pole1; //domyślnie prywatna
public: //działa na wszystkie składowe poniżej
    int pole2;
    int metoda1();
private: //dopóki nie użyjemy kolejnego specyfikatora
    int metoda2();
}; //musi być zakończona średnikiem
```

Domyślnie składowe klasy są prywatne. Oznacza to, że są dostępne tylko wewnątrz tej klasy. Aby to zmienić należy używać specyfikatorów dostępu. Dostępne są trzy specyfikatory: `public`, `protected i private`. Specyfikatory działają na składowe klasy wymienione po nim. Specyfikator `protected` ma znaczenie przy dziedziczeniu i zostanie omówiony później. Specyfikator `public` oznacza, że składowe po nim wymienione są dostępne wszędzie – w klasie i poza nią. Zamiast słowa kluczowego `class` można w C++ zastosować słowo `struct` (w C++ „struktury” mogą też mieć metody). Wtedy wszystkie składowe domyślnie są publiczne.

Ciało metody może być zdefiniowane wewnątrz klasy:

```
class Klasa {
    int pole1;
public:
    int pole2;
    int metoda1() //wewnątrz deklaracji klasy
    {
        std::cout<<"metoda1"<<std::endl;
        return 1;
    }
private:
    int metoda2();
};
```

lub poza deklaracją klasy:

```
int Klasa::metoda2() //poza deklaracją klasy trzeba podać nazwę klasy
{
    std::cout<<"metoda2"<<std::endl;
    return 2;
}
```

Klasy najczęściej służą jako swego rodzaju szablon do tworzenia obiektów (egzemplarzy, instancji klasy). Obiekt tworzy się tak jak każdą inną zmienną np.:

```
    Klasa obiekt1;
    Klasa obiekt2;
```

Wywołanie metody na stworzonym obiekcie odbywa się poprzez odwołanie do niej za pomocą kropki:

```
    obiekt1.metoda1();
    obiekt2.metoda1();
```

Podobnie można się odwołać do pola np.

```
    obiekt1.pole2=1;
```

Konstruktor

Konstruktor jest wywoływany zawsze, gdy tworzymy obiekt danej klasy. Jeśli klasa nie zawiera konstruktora, kompilator automatycznie tworzy konstruktor, który nic nie robi. Konstruktor nie pojawi się nigdzie w kodzie, jednak będzie on istniał w skompilowanej wersji programu i będzie wywoływany za każdym razem, gdy będzie tworzony obiekt klasy. Jeśli chcemy zmienić domyślne własności konstruktora i nadać polom w obiekcie wartości początkowe, należy utworzyć własny konstruktor dla klasy. Konstruktor nie ma typu i jego nazwa jest ZAWSZE identyczna z nazwą klasy. Można tworzyć kilka konstruktorów dla jednej klasy (muszą się one jednak różnić liczbą lub typem argumentów). Przykład:

```
class Klasa {
    //...
public: //konstruktor najczęściej jest publiczny
    Klasa(int p1, int p2) //ciało konstruktora może być zdefiniowane wewnątrz
lub na zewnątrz klasy (jak ciało metody)
    {
        pole1=p1;
        pole2=p2;
    }
    //...
};
```

Poza nazwą i brakiem typu konstruktor definiuje się tak samo jak metodę. Parametry konstruktora przekazuje się w momencie tworzenia obiektów.

```
    Klasa obiekt1(1,2);
    Klasa obiekt2(3,4);
```

Zadanie 1

Zdefiniuj klasę Zwierze. Powinna ona zawierać:

- prywatne pole `mImie` typu `string`
- konstruktor ustawiający imię zwierzęcia (imię jest parametrem konstruktora)
- publiczną metodę `ustawImie(string imie)` zmieniającą imię zwierzęcia
- publiczną metodę `naLancuch()`, która zwraca stałą referencję do pola imię.

W funkcji `main()`

- stwórz i wyświetl (używając metody `naLancuch()`) 2 obiekty klasy `Zwierze`.
- Wypróbuj metodę ustawImie()

Zadanie 2

Zdefiniuj funkcję `porownaj(z1,z2)`, która jako parametry otrzymuje obiekty reprezentujące zwierzęta, porównuje pola `mImie` i zwraca `true` jeżeli pola są takie same, a `false` w przeciwnym wypadku.

Składowe statyczne

Składowe statyczne poprzedza się słowem kluczowym `static`. Składowe te nie są związane z instancjami klas (obiektami). Aby ich używać nie trzeba tworzyć obiektów. Metody statyczne mają dostęp tylko do składowych statycznych. Metody niestatyczne mają dostęp do składowych statycznych. Poniżej przedstawiono przykład tworzenia i wykorzystania składowych statycznych.

```
class Klasa2 {
public:
    static int x;
    static void metodaStatyczna()
    {
        std::cout<<"metoda statyczna"<<std::endl;
    }
};

int main(int argc, char *argv[])
{
    std::cout<<Klasa2::x<<std::endl;
    Klasa2::metodaStatyczna();
}
```

Pola statyczne można traktować jak pola wspólne dla wszystkich obiektów klasy.

Zadanie 3

W klasie `Zwierze`:

- dodaj pole statyczne `licznik`, które będzie przechowywało liczbę utworzonych zwierząt
- zmodyfikuj konstruktor aby zliczał utworzone obiekty
- dodaj metodę statyczną `ileZwierzat()`, która zwraca liczbę aktualnie istniejących zwierząt