Egyszerű osztály

Az egyszerű osztály elsősorban egy felhasználói típus definiálására szolgáló nyelvi eszköz. Adattagjainak helyfoglalása statikus módon történik (`new` nélkül).

Osztály és metódus definíció

```
 // osztály definíció
class O
{
   public:
   // metódusok deklarációi
      O();  // konstruktor
      ~O(); // destruktor
      Tipus11 Metod1();
      Tipus12 Metod2();
      ...

   private:
   // adattagok
      Tipus21 tag1;
      Tipus22 tag2;
      ...
   // metódusok deklarációi
      Tipus31 BelsőMetod1(…);
      Tipus32 BelsőMetod2(…);
      ...
};

 // metódusok definíciói
O::O(){…}
O::~O(){…}
Tipus11 O::Metod1(){…}
Tipus12 O::Metod2(){…}
...
Tipus31 O::BelsőMetod1(…);
Tipus32 O::BelsőMetod2(…);
...
```

Egy osztály-definíció után már tudunk osztály típusú változókat, úgynevezett objektumokat definiálni. Ezekre az objektumokra mindig végrehajtódik az osztály konstruktora. Ha a konstruktor paramétereket igényel, akkor azokat az objektum definíciójánál meg kell adni.

```
O o(…);
```

Az osztály-definíció után az osztály publikus részben deklarált függvényeket, az úgynevezett metódusokat meghívhatjuk az osztály egy adott objektumára. Egy metódus hívásakor a metódusneve elé kell írni azt az objektumot (ez a metódus alapértelmezett objektuma), amelyre a metódust meghívjuk.

```
o.Metod1(…)
```

A privát rész metódusai viszont csak ugyanazon osztály más metódusaiból hívhatóak meg. Az ilyen „belső” metódushívásnál nem kell feltűntetni az alapértelmezett objektumot, az automatikusan a hívó metódus alapértelmezett objektuma lesz.

Az osztályban deklarált metódusokat definiálni kell. Ezek – mint azt az előbb láttuk – abban különböznek a szokásos függvényektől, hogy mindegyiknek alapértelmezés szerinti paramétere az osztály egy objektuma. Azt az objektumot, amelyre a metódust meghívták, az adattagok reprezentálják, és az adattagok pillanatnyi értékei határozzák meg az objektum értékét. A metódus törzsében ennek az objektumnak az értékéhez az azt reprezentáló adattagokon keresztül, közvetve lehet hozzáférni úgy, hogy az adattagok nevei önálló változónevekként jelennek meg a függvénytörzsben. Ha a metódus más objektumokat is használ, akkor azok adattagjaira a `struct` szerkezetnél látott módon hivatkozhatunk; az adattag nevei mezőnevekként jelennek meg az objektum neve után.

```
o.tag1
```

Programkódunk más részein, a metódusok definícióján kívül, semmilyen formában sem lehet használni a privát adattagokat.

Osztályok elhelyezése a programkódban

Kétféle elhelyezést engedünk meg.

Az első az, amikor az osztály szolgáltatásait (osztályt, mint típusnevet, valamint az osztály publikus metódusait) használó forrásállomány elején definiáljuk az osztályt, és ugyanebben a forrásállományban az egyéb függvény-definíciók között definiáljuk az osztály metódusait.

A másik az, amikor egy osztály-definíciót egy külön fejállományban, az ahhoz tartozó metódus-definíciókat egy ugyanilyen nevű forrásállományban helyezzük el, és a fejállományt „beinklúdoljuk” mind a metódus-definíciókat tartalmazó forrásállományba, mind az összes olyan állományba, ahol az osztály szolgáltatásait használni akarjuk.

Konstans metódusok

Olyan metódusok, amelyek nem változtatják meg az alapértelmezett objektum adattagjait. Ezt a formális paraméter lista mögött elhelyezett **`const`** szó jelzi. A fordító program ellenőrzi.

Inline metódus definíció

A metódusnak az osztálydefinícióban történő definiálása, amit a metódus deklarációjának helyébe írunk. Csak egyszerű, néhány értékadásból álló függvénytörzs esetén alkalmazzuk.

Konstruktorok és destruktor

Minden osztálynak vannak speciális metódusai, az osztály nevével megegyező konstruktorai illetve ezek ellentét párja, a destruktor. (Ha elfelejtenénk definiálni ilyeneket, akkor is lesz egy alapértelmezett konstruktor illetve destruktor.) A konstruktorok feladata az adott objektumot reprezentáló adattagok létrehozása, a destruktoré azok megszűntetése. Egyszerű osztályok esetében a konstruktor csak kezdeti értéket ad az adattagoknak. Egy osztálynak több (paraméterezésükben eltérő) konstruktora is lehet. Ezek törzsében olyan kódot írhatunk, amely az általunk megadott módon inicializálja az adattagokat. Különösen előnyös ez akkor, amikor egy típus osztállyal történő megvalósításánál a típusértéket (az objektum értékeket) reprezentáló adattagokra teljesülnie kell egy invariáns állításnak, amit már az objektum létrehozásakor érvényesíteni akarunk.

Destruktorból csak egy van. Egyszerű osztályok készítésénél csak akkor definiálunk destruktort (különben az alapértelmezett destruktor van érvényben), amikor a konstruktorban olyan inicializáló utasítást használtunk, amelynek van befejező párja. (Például fájloknál az open-close utasításpár.)

Adattagok inicializálása a konstruktor fejében

A konstruktorok definíciójában a formális paraméter listát követő kettőspont után az alapértelmezett objektum adattagjainak konstruktorát hívhatjuk meg

```
O::O(): tag1(…), tag2(…), … {…}
```