

A Programozás Alapjai 2

Objektumorientált szoftverfejlesztés

Dr. Forstner Bertalan
forstner.bertalan@aut.bme.hu



Department of
Automation and
Applied Informatics

Behelyettesíthetőség, virtuális függvények

- **Kérdés:** hogyan tároljunk el kör és téglalap típusú objektumokat egy tömbbe?
- **Válasz:** Behelyettesíthetőség. Mi is az?

- **A C++-ban a változó típusában csak egy megfelelő típusú változó lehet.**

- A C++-ban a változó típusában csak egy megfelelő típusú változó lehet.

```
int a;  
double b=3.14;  
a=b;
```

- Ettől *a* nem lett *double* típusú, sem *b* *int* típusú.
- Csak meghívódott egy konverziós függvény a kettő között.

- Ez így van osztályok között is:

```
class A{...} ;
```

```
class B: public A{...} ;
```

```
B b ;
```

```
A a=b ;
```

- Ezután *b* marad *B* típusú, és *a* marad *A* típusú.
- A konstruktor működésétől függ az eredmény.

- A pointer esetén már két paramétertől függ
- Példa

- A pointer esetén már két paramétertől függ
- Példa

```
double a = 0;  
char*p = (char*)&a;  
(*p)++;  
cout << a; //4.94066e-324
```


TV és nem hozzá való távirányító...



TV és nem hozzá való távirányító...

- A fogantyú határozza meg a műveleteket
- A benne tárolt érték és a fogantyú teljesen különálló
- **polimorfizmus, polimorf mutató**

TV és nem hozzá való távirányító...

- A fogantyú határozza meg a műveleteket
- A benne tárolt érték és a fogantyú teljesen különálló
- **polimorfizmus, polimorf mutató**
- Ez így OK: **Az őosztály típusú fogantyú mögött baj nélkül lehet leszármazott típusú objektum!**
 - > Persze ezen keresztül csak a leszármazottra érvényes részt érjük el.
- Ez a referenciára is igaz.

Példa

- Person,
- Belőle származó Employee és Student
- isfrom20Century függvény
 - > Nem fogja tudni, hogy a paramétere Person vagy valamelyik leszármazottja

Példa 2

- Írjunk egy alkalmazás részletet egy rajzoló programhoz!
- A felhasználó geometriai alakzatokat hoz létre
- Mi ezeket szeretnénk tárolni közös tömbben
- A megoldás

Virtuális függvények

- A virtuális azt jelenti, ha
 - > őssztályban virtuálisnak definiálok egy függvényt,
 - > és a leszármazottban létezik ugyanolyan névvel, paraméterekkel,
 - > és a leszármazotton ezt a függvényt egy őssztály típusú pointeren keresztül hívjuk meg,
 - > akkor a leszármazottbeli tag fog meghívódni.

Heterogén kollekció

- A példában a shapes tömb
- Ősosztály típusú pointereket tárol
- De ezek leszármazottakra mutatnak

A pair of red theater curtains with gold tassels, partially drawn to reveal the word "Intermission" in a white, elegant script font. The background is black, suggesting a stage or theater setting.

Intermission

Tehát: Virtuális függvények

- A virtuális azt jelenti, ha
 - > őssztályban virtuálisnak definiálok egy függvényt,
 - > és a leszármazottban létezik ugyanolyan névvel, paraméterekkel
 - > és a leszármazotton ezt a függvényt egy őssztály típusú pointeren keresztül hívjuk meg,
 - > akkor a leszármazottbeli tag fog meghívódni.
- **Akkor most vezessük be a területet! (Példa)**

Absztrakt osztály

- Tisztán virtuális függvény (pure virtual)
- **virtual double Area() = 0;**

Korlátozó öröklés

- Be szeretnénk vezetni a **Square** osztályt. hol lesz a **helye**?
 - > Rectangle és Shape között, hisz csak 1 oldalparamétere van?
 - > Nem túl elegáns...
- Példa: A **Square** bevezetése

Destruktor kérdése

- Melyik destruktor hívódik meg?
 - > Miért, virtuális?

Destruktor kérdése

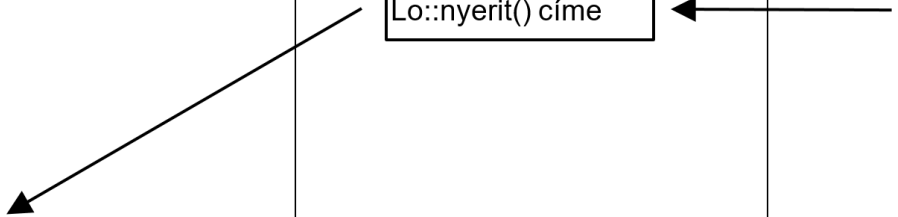
- Melyik destruktor hívódik meg?
 - > Miért, virtuális?
- Ha egy osztályból **várhatóan leszármazunk**, vagy **van virtuális függvénye**, akkor legyen a **destruktora is virtuális**.
 - > (Hisz **őosztály pointerén** keresztül hívjuk meg a **delete-t.**)
 - > Ha nem ezt tesszük, akár le is állhat az alkalmazás, de mindenképpen számítani kell memóriaszivárgásra!

Virtuális függvénytábla







- A virtuális függvények megvalósításának alapja az indirekció.
- Az a függvény cím, amire meghíváskor ugrani kell, nem fordításkor dől el, hanem futás közben

Virtuális függvénytábla

- Minden **osztály** rendelkezik a memóriában egy ugrótáblával az összes virtuális függvényére
 - > akár benne definiált, akár örökölt függvényről van szó
 - > Annyi címet tartalmaz ez a tábla, ahány virtuális függvénye van az adott osztálynak
- Minden **objektum** rendelkezik egy **vfptr** (vagy hasonló) nevű pointerrel az osztályának virtuális ugrótáblájára.
- (megvalósítás-függő)

| Osztály kód | Osztály ugrótábla | Objektum memóriakép |
|---|---|--|
| <pre>class Lo { string nev; public: void uget(); virtual void nyarit(); };</pre> | <div>Lo::nyarit() címe</div>  | <div>Lo lo1;</div> <div><div>vfp ptr</div><div>nev</div></div> |

| Osztály kód | Osztály ugrótábla | Objektum memóriakép |
|--|------------------------------|--|
| <pre>class Lo { string nev; public: void uget(); virtual void nyerit(); };</pre> | <div>Lo::nyerit() címe</div> | <div>Lo lo1;</div> <div><div>vfptr</div><div>nev</div></div> |
| <pre>class HidegveruLo : public Lo { int verHomerseklet; public: //Lo* pointererel nem latszik void uget(); };</pre> | <div>Lo::nyerit() címe</div> | <div>HidegveruLo lo2;</div> <div><div>vfptr</div><div>nev</div><div>verHomerseklet</div></div> |

| Osztály kód | Osztály ugrótábla | Objektum memóriakép |
|--|---|--|
| <pre>class Lo { string nev; public: void uget(); virtual void nyerit(); };</pre> | <div data-bbox="1039 189 1309 247">Lo::nyerit() címe</div>  | <div data-bbox="1566 147 1657 175">Lo lo1;</div> <div data-bbox="1561 189 1787 304"> <div data-bbox="1657 204 1715 232">vfptr</div> <div data-bbox="1657 261 1715 289">nev</div> </div>  |
| <pre>class HidegveruLo : public Lo { int verHomerseklet; public: //Lo* pointerrel nem latszik void uget(); };</pre> | <div data-bbox="1039 575 1309 632">Lo::nyerit() címe</div>  | <div data-bbox="1528 532 1754 561">HidegveruLo lo2;</div> <div data-bbox="1522 575 1748 746"> <div data-bbox="1619 589 1676 618">vfptr</div> <div data-bbox="1619 646 1676 675">nev</div> <div data-bbox="1532 704 1744 732">verHomerseklet</div> </div>  |
| <pre>class Unikornis : public HidegveruLo { int nyelvKod; public: void nyerit(); virtual void beszeli(); };</pre> | <div data-bbox="967 961 1348 1075"> <div data-bbox="967 975 1329 1003">HidegveruLo::nyerit() címe</div> <div data-bbox="967 1032 1335 1061">HidegveruLo::beszeli() címe</div> </div>  | <div data-bbox="1528 918 1715 946">Unikornis lo3;</div> <div data-bbox="1522 961 1748 1189"> <div data-bbox="1619 975 1676 1003">vfptr</div> <div data-bbox="1619 1032 1676 1061">nev</div> <div data-bbox="1532 1089 1744 1118">verHomerseklet</div> <div data-bbox="1580 1146 1705 1175">nyelvKod</div> </div>  |

