

Programozás II.

2. gyakorlat

Követelmények

- 2 kötelező program (**Igen/Nem**)
 - Előre megadott listából kap majd mindenki feladatot
 - Első feladat
 - *Beadás:* október 15. 08:00, Coospace
 - Második feladat
 - *Beadás:* november 19. 08:00, Coospace
- Gyakorlati vizsga (**100 pont / 50 pont**)
 - Egy megadott program elkészítése (2 óra)
 - Utolsó két hétben legalább egyszer el kell menni
 - Egy javítási lehetőség, minimum pontszámért

Objektum-orientált programozás

- Programozási paradigma, amelyben a programot objektumokból építjük fel. A program működése objektumok kommunikációját jelenti.
- Alapelvei:
 - egységbezárás
 - öröklődés
 - polimorfizmus

Objektum-orientált programozás

- **Osztály:** Leírás objektumok csoportjához, melyeknek közösek az
 - attribútumai, operációi
 - más objektumokkal való kapcsolataik
 - viselkedésük
- az osztály az objektum típusa
- lehetséges állapotok halmaza

Objektum-orientált programozás

- **Objektum:** Információt tárol, és kérésre műveleteket végez. Van állapota, viselkedése, futásidőben azonosítható.
- az objektum az osztály példánya
- egy a lehetséges állapotok közül

Objektum-orientált programozás

Class

Definition of objects that share structure, properties and behaviours.



Building
class



Dog
class



Computer
class

Instance

Concrete object, created from a certain class.



Empire State
instance of Building



Lassie
instance of Dog



Your computer
instance of Computer

Egységbezárás, öröklődés, polimorfizmus

- **Egységbezárás:** Az adatok és rajtuk végzett műveletek egységbezárása osztályokba. Az objektum egységbe zárja az állapotot (adattagok értéke) a viselkedésmóddal (metódusok).
- **Öröklődés:** Osztályok között értelmezett viszony, egy általános típusból egy specializáltabb típust tudunk létrehozni (ősosztály, gyerekosztály). A gyerekosztály mindent örököl az őstől, azt saját adatokkal és műveletekkel egészíthetjük ki.
- **Polimorfizmus:** Többalakúság, objektumok felcserélhetőségét biztosítja; egy típust az őse alapján kezeljük

Overloading/Overriding

- **Metódus túlterhelés (overloading):** Azonos nevű, de különböző paraméterlistával rendelkező függvények.
- **Metódus felüldefiniálás (overriding):** A gyerekosztályban felüldefiniálhatjuk az ősosztályban definiált függvényt.
 - Ha ősosztály típusú mutatón vagy referencián keresztül érjük el az osztályhierarchia példányait és ezen keresztül meghívjuk a felülírt metódust, akkor futási időben dől el, hogy pontosan melyik metódus kerül meghívásra. (kései kötés / late binding).

Operátorok - Aritmetikai műveletek

- $kif1 + kif2$: összeadás
- $kif1 - kif2$: kivonás
- $kif1 * kif2$: szorzás
- $kif1 / kif2$: osztás
- $kif1 \% kif2$: maradékos osztás
- $- kif$: ellentett képzés

Operátorok - Logikai műveletek

- `kif1 && kif2` : logikai és
- `kif1 || kif2` : logikai vagy
- `! kif` : logikai tagadás
- Például:

```
int a=8;  
if( ( a%10 == 8 ) || (2*a > 10) )  
    printf("az a megfelelő");
```

Operátorok - Relációs műveletek

- $kif1 == kif2$: egyenlő
- $kif1 != kif2$: nem egyenlő
- $kif1 < kif2$: kisebb
- $kif1 > kif2$: nagyobb
- $kif1 \leq kif2$: kisebb-egyenlő
- $kif1 \geq kif2$: nagyobb-egyenlő
- Short-circuit evaluation
 - https://en.wikipedia.org/wiki/Short-circuit_evaluation

Operátorok - bitszintű műveletek

- $\sim \text{kif}$: komplement képzés (bitenkénti negálás)
- $\text{kif1} \gg \text{kif2}$: bitléptetés jobbra
- $\text{kif1} \ll \text{kif2}$: bitléptetés balra
- $\text{kif1} \& \text{kif2}$: bitenkénti és művelet
- $\text{kif1} | \text{kif2}$: bitenkénti vagy művelet
- $\text{kif1} \wedge \text{kif2}$: bitenkénti kizáró vagy

Operátorok - Értékadó műveletek

- változó = kif - értékadás
- +=, -=, *=, /=, %= - művelettel egybekötött értékadás
- >>=, <<=, &=, ^=, |= - művelettel egybekötött értékadás
- Például:
int x = 8;
x += 2; //x = 10

Operátorok - Inkrementálás

- Prefix operátorok
++kif, --kif : kif inkrementálása (dekrementálása)
visszatérési értéke a növelt érték
- Postfix operátorok
kif++, kif-- : kif inkrementálása (dekrementálása)
visszatérési értéke a növelés előtti érték (!)

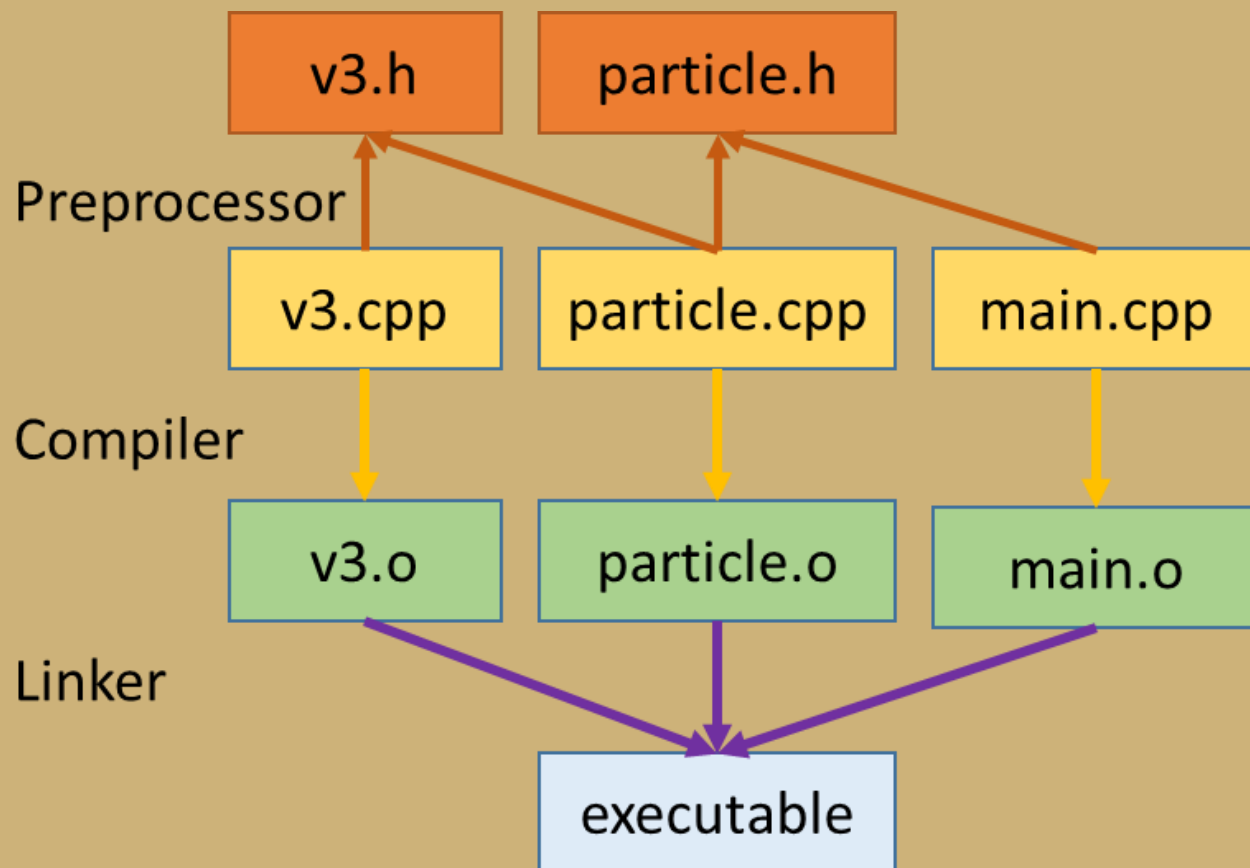
Például:

```
int x = 0, y = 1;
```

```
x = ++y;           //x = 2, y = 2
```

```
x = y++;           //x = 2; y = 3
```

Fordítás



Fordítóprogramok

- Microsoft
 - Visual Studio, cl.exe
- GNU
 - gcc, g++
- LLVM
 - clang, clang++

Build rendszerek

- Fordítás, linkelés
- Make
 - Programozás alapjai
- CMake
 - ingyenes, nyílt forrású
 - <https://cmake.org/cmake-tutorial/>
- Ninja
 - <https://ninja-build.org/>