A Programozás Alapjai 2
Objektumorientált szoftverfejlesztés

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

Kitérő: tömb inicializálás

```
int array1[3];
printf("%d\n", array1[0]); //random szemet

int array2[3] = {}; //zero-inicializalas
printf("%d\n",array2[0]);

int array3[3] = {1,2}; //a maradek zero
printf("%d\n", array3[0]);

int array4[3] { 1,2,3 }; //a c++11 szabvany ota
printf("%d\n", array4[0]);
```

Dinamikus tagváltozók

A malloc hátránya

- A *malloc* nem hívja meg a konstruktorokat, nem tudunk a szintaxis miatt paramétereket átadni
- Új nyelvi elem: *new*, *delete*

- Példa

new / delete

```
int* i = new int;
*i = 5;
…
delete i;

int * ii = new int[100];
…
ii[50] = 50;
…
delete[] ii;
```

New használata list initializationnel

- Természetesen lehet CPP’11 óta

```
int* ptr{ new int { 42 } }; //ptr egy int-re mutat a heap-en, aminek értéke 42

//array egy „length” hosszú int tömbre mutat a heapen
int* array{ new int[length]{} };

//inicializálással, a maradék 0
int* array{ new int[5] {2,1} };
// ”2, 1, 0”
printf("%d, %d, %d\n", array[0], array[1], array[2]);

//Pointerek pointere: Foglaljunk egy tömböt 10 int pointernek
int** array2 { new int*[10] };
```

Másoló konstruktor

- Feladata az objektum inicializálása egy másik, ugyanolyan osztálybeli objektum alapján.
  > Ha nem írok, **létrejön egy alapértelmezett**, ami bitről bitre másol.
- Milyen esetekben kell megírni?
  > Ha ez a viselkedés nem felel meg nekünk. Például a legtipikusabb eset:
  > Amikor az adott **objektum** maga **felelős** a valamilyen hozzá tartozó dinamikusan allokált (malloc/free, new/delete) **memória lefoglalásáért** és **felszabadításáért**.
- Példa

Példa probléma

```
class Person
{
private:
   char* name;

public:
   Person(char* nameparam)
   {
      name = new char[strlen(nameparam) + 1];
      strcpy_s(name, strlen(nameparam) + 1, nameparam);
   }
   ~Person() {
      delete[] name;
   }
};
```

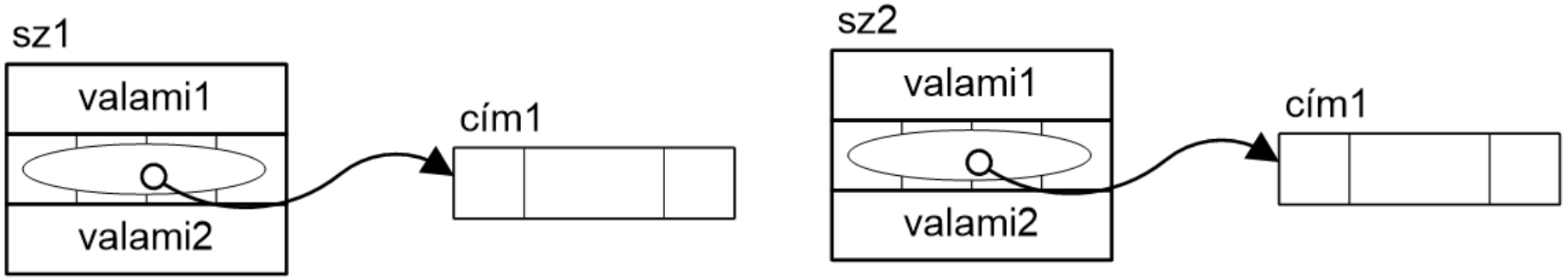
```
int main(int argc, char* argv[]) {

  Person pisti("Pisti");
  Person iker(pisti);

  getchar();
  return 0;
}
```

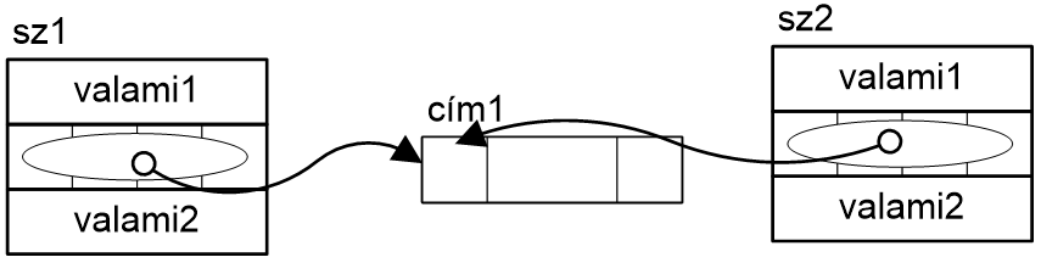
A Person szemléltetése memóriaképpel

- A probléma: a bitről bitre másolás inkonzisztenciához vezet.
- Mit várunk el a másolat létrejöttekor?



A Person szemléltetése memóriaképpel

- Mit kapunk helyette:



- Mostantól ketten felelősek ugyanazon memória terület menedzseléséért. Probléma pl. a következő:
  - > Amikor az sz1 (pisti) felszabadul (main-ből kilépéskor) meghívódik a destruktora, ami felszabadítja az objektumhoz tartozó, általa dinamikusan lefoglalt memória területet (név).
  - > Amikor az sz2 (iker) megszűnik, neki is meghívódik a destruktora, ami felszabadítaná a már felszabadított területet: durva futás idejű hiba!
  - > Ugyanígy hiba lenne: ha módosítjuk az egyik stringet, a másik is módosul.

Másoló konstruktor írás

- A **megoldás**: mivel a **default másoló nem megfelelő**, felül kell írni és a megfelelő viselkedést le kell programozni.

- Példa

Másoló konstruktor írás

- A **megoldás**: mivel a **default másoló nem megfelelő**, felül kell írni és a megfelelő viselkedést le kell programozni.
- Példa

```
Person(const Person& other)
{
  name = new char[strlen(other.name) + 1];
  strcpy_s(name, strlen(other.name) + 1, other.name);
  printf("Copy %s!\n", name);
}
```

Másoló konstruktor írás

- **Mikor hívódik meg** a másoló konstruktor? Amikor csak másolat készül.
  - > **explicit**, mint a példában
  - > függvénynek **paraméterként** átadva (nem referencia vagy pointer)
    - Példa: kick függvény
  - > **visszatérési érték** függvényben
  - > **bonyolult** kifejezésekben temporális változóként

Mivel sok esetben szükséges, mindig írjunk másoló konstruktort, ha az osztály objektumaihoz dinamikusan lefoglalt terület tartozik, aminek kezeléséért maga az osztály felelős.

Mikor hívódik konstruktor és destruktor?

- Példa

Miért referencia a copy constructor paramétere?

- Különben végtelen ciklus lenne. Miért is?

- A VS nem is engedi, próbáljuk ki!

Kompozíció vs. Aggregáció

- A birtokolja B-t: kompozíció. B-nek semmi értelme, létcélja nincs a rendszerben A nélkül.
  > Például: **Személynek** van **neve**.

- A “használja” B-t: aggregáció. B koncepcionálisan teljesen függetlenül létezik A-tól.
  > Például: **Személynek** van **apja**, aki egy másik Személy, de független.

Egy összetett példa

- IntFifo