Memóriakezelés C++-ban

Pataki Norbert

Programozási Nyelvek és
Fordítóprogramok Tanszék

Programozási Nyelvek I.

Témák

Memória szegmensek

- Statikus tárterület
  - data szegmens
  - bss szegmens
- code szegmens
- Runtime stack, végrehajtási verem, stack szegmens
- Heap

Dinamikus memória allokáció – heap

```
int *p = 0;
int k;
/* ... */
p = (int*)malloc( k * sizeof( int ) );
...
free( p );
```

Melyik pointert szabadítjuk fel?

- ```
  int s = 3;
  int* p = &s;
  ```
- ```
  char *m = "Hello";
  ```
- ```
  int a[] = {3, 2, 8};
  int *ap = a;
  ```
- ```
  int* dp = (int*)malloc(s * sizeof( int ) );
  ```

Melyik pointert szabadítjuk fel?

- ```
  int s = 3;
  int* p = &s;
  ```
- ```
  char *m = "Hello";
  ```
- ```
  int a[] = {3, 2, 8};
  int *ap = a;
  ```
- ```
  int* dp = (int*)malloc(s * sizeof( int ) );
  ```
- ```
  free( dp );
  ```
  ```

Dinamikus memóriakezelés C++-ban

```
int x;
std::cin >> x;
int *p = new int[ x ];
// ...
delete [] p;
```

Dinamikus memóriakezelés C++-ban

```
int s;
std::cin >> s;
int *q = new int( s );
// ...
delete q;
```

Problémák

- Memória szivárgás (memory leak)
- `malloc`/`free`, `new`/`delete`
- Dupla `free`, `delete`
- `new[]`/`delete`, `new`/`delete[]`
- `new`, `delete`: élettartam szabályok
- `malloc`, `free`: csak memóriakezelés

Probléma

```
void f()
{
  int x;
  std::cin >> x;
  int *p = new int[ x ];
  // ... Kivétel (exception) esetén: :-(
  delete [] p;
}
```

Ötlet

- Csomagoljuk be egy osztályba:
  - konstruktor: élettartam elején kötelezően lefut
  - destruktor: élettartam végén kötelezően lefut (akár kivétel (exception) miatt)
  - Például: `std::vector`
  - Használjuk automatikus saját típusú változóként (objektum)
- Bjarne Stroustrup: „Erőforrás-orientált programozási nyelv”
- Nem csak memória: erőforrás mindenhol – file-ok, stb.
- Erőforrás-kezelés != Másolás

Resource acquisition is initialization (RAII)

C++ Standard Template Library (STL) konténerek

- Szekvenciális konténerek:
  - `std::vector`
  - `std::string`
  - `std::list`
  - `std::deque`
- Asszociatív konténerek:
  - `std::set`, `std::multiset`
  - `std::map`, `std::multimap`