Dinamički životni vek

- Izraz (operator) *delete* ima jedan od sledeća dva oblika:

```
delete expression
delete [] expression
```

- Prvi oblik se koristi ako se uništava jedan objekat; drugi oblik je obavezan ako se uništava niz objekata. Ako se u ovome pogreši, efekat je nedefinisan (greška u izvršavanju, jer se pozivaju pogrešne operatorske funkcije za dealokaciju)
- Izraz koji je operand ovog operatora mora biti tipa pokazivača na objektni tip. On mora ukazivati na objekat, odnosno niz objekata napravljen pomoću *new*, ili na podobjekat osnovne klase objekta napravljen pomoću *new*, ili imati vrednost *null* (u kom slučaju ovaj operator nema efekta); u suprotnom, efekat je nedefinisan
- Izraz (operator) *delete* uvek radi sledeće stvari (osim ako pokazivač ima *null* vrednost, kada nema efekta), ovim redom:
    1. poziva destruktor objekta na koji pokazivač ukazuje, ili destruktor svakog elementa niza, ukoliko je pokazivač na klasni tip; ako pokazivač ukazuje na podobjekat osnovne klase, a destruktor je virtuelan, poziv je polimorfan; ako pokazivač ukazuje na podobjekat osnovne klase, a destruktor nije virtuelan, ponašanje je nedefinisano
    2. oslobađa (dealocira) prostor koji je zauzimao objekat, odnosno niz, pozivom odgovarajuće operatorske funkcije za dealokaciju
- Operator *delete* uvek ima povratni tip *void*
- Potpuno analogno alokaciji, delokacija prostora (drugi korak) vrši se pozivom neke od (preklopljenih) operatorskih funkcija koje su standardno definisane jezikom (postoje i ovakve operatorske funkcije sa još nekim parametrima) i koje se mogu zameniti ili redefinisati za klase:

```
void operator delete   (void* ptr);
void operator delete[] (void* ptr);
```

Dinamički životni vek

- Tzv. *curenje memorije* (*memory leak*) je problem koji može nastupiti nekorektnim rukovanjem dinamičkim objektima, tako što se dinamički objekti repetitivno prave, ali se ne uništavaju, jer je programer zaboravio da propisno uništava dinamičke objekte koji više nisu potrebni (a program iznova pravi nove kada su potrebni)
- Nakon dužeg izvršavanja programa, slobodna memorija će biti iscrpljena, pa memorije za nove dinamičke objekte više neće biti, i naredna operacija *new* baciće izuzetak (ili vratiti *null* vrednost); nakon toga, program više neće radi kako se od njega očekuje
- Na primer, sledeći deo koda je banalan, ali i očigledan primer curenja memorije:

```
int i = *new int(0);
```

  - Tipičan uzrok jeste nepažljivo dodeljena odgovornost za brisanje dinamički napravljenih objekata. Na primer, korisniku neke funckije koja pravi nov dinamički objekat i vraća pokazivač (ili referencu) na njega nije jasno da je odgovornost za brisanje tog objekta na njemu (a ne na onom ko ga je koristio), pa će zaboraviti da ga obriše:

```
extern X* getAnX (…);
X* px = getAnX(…);
```

  - Uzrok može biti i situacija u kojoj se objekat ne obriše zbog izuzetka:

```
void f () {
  int* p = new int(1);
  g();
  delete p;
}
```