Dinamički životni vek

- Izraz *new* vraća vrednost koja je po tipu pokazivač na tip specifikovan pomoću *type-id* i koji ukazuje na objekat napravljen ovim izrazom, odnosno na prvi objekat u nizu, ako je napravljen niz
- Izrazom *new* mogu se praviti objekti (ne i reference), pošto ne postoje pokazivači na reference (pa operator *new* ne može da vrati pokazivač na dinamički kreiranu referencu)
- Ova vraćena vrednost je jedina veza do napravljenog dinamičkog objekta koji je inače anoniman (nema ime); ukoliko se ova veza izgubi, objekat ostaje nedostupan, pa se ne može ni uništiti:

```
int i = new int(2);
int j = *new int(2);
int& r = *new int(2);
delete &r;
```

Greška u prevođenju: ne postoji implicitna konverzija iz *int** u *int*

Problem: pristup do dinamičkog objekta je trajno izgubljen

Referenca upućuje na dinamički objekat, pa je ona sada njegovo ime

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);
```