Dinamički životni vek

- Opcioni argumenti za parametre smeštanja mogu se koristiti za prosleđivanje dodatnih informacija (parametara) operatorskim funkcijama koje izraz *new* poziva za potrebe alokacije prostora za dinamički objekat. Dve standardno podržane mogućnosti za ove parametre su sledeće:
  - parametar za tzv. *placement new* (po kom su ovi parametri i dobili ime, jer su uvedeni u jezik kao uopštenje ove mogućnosti)
  - argument *nothrow*
- Tzv. *placement new* omogućava da se dinamički objekat smesti u prostor već alociran za neki drugi objekat, odnosno u skladište koje još uvek traje. Na primer:

```
char* ptr = new char[sizeof(T)];
```
Alokacija prostora za niz znakova date dimenzije

```
T* tptr = new(ptr) T;
```
Placement new: pravi se objekat tipa *T* u prostoru na koji ukazuje argument *ptr*

```
tptr->~T();
```
Objekat tipa *T* se uništava eksplicitnim pozivom destruktora

```
delete [] ptr;
```
Dealokacija prostora

- Podrazumevano, izraz *new*, tačnije, operatorska funkcija koju on poziva za alokaciju prostora baca izuzetak ukoliko ne može da alocira taj prostor; ako se kao argument dostavi predefinisani objekat *std::nothrow*, neće biti bačen izuzetak, nego će izraz *new* vratiti *null* vrednost u ovom slučaju:

```
auto p = new (std::nothrow) (Clock[2*n][Max]) {}
if (p) …p[i][j]…
```

Dinamički životni vek

- *Placement new* zapravo omogućava inicijalizaciju objekta pozivom konstruktora za deo memorije koji je već određen za smeštanje objekta; drugačije nije moguće pozvati konstruktor koji inicijalizuje objekat na definisanom mestu u memoriji
- *Placement new* se može koristiti u sistemskim rutinama niskog nivoa, recimo u delovima sistema (npr. operativnog sistema) koji upravljaju alokacijom memorije, keširanjem blokova sa diska, implementaciji fajl sistema i slično
- Na primer, želimo da dealocirane segmente memorije uvezujemo u listu, s tim da strukturu *FreeSeg* koja predstavlja jedan slobodan segment smestimo baš na početak tog slobodnog segmenta (a ne da alociramo poseban deo memorije koji bi se trošio za evidenciju slobodne memorije):

```
class FreeStore {
public:
  inline void free (void* addr, size_t sz);
  …
private:
  class FreeSeg {
  public:
    FreeSeg (FreeSeg* nxt, size_t sz) : next(nxt), size(sz) {}
  private:
    FreeSeg* next;
    size_t size;
  };
  FreeSeg* head;
};
void FreeStore::free (void* addr, size_t sz) {
  head = new (addr) FreeSeg(head,sz);
}
```