Konstantna inicijalizacija

- Na pimer, *enumeratorske klase*, tj. klase sa konstantnim skupom instanci predstavljaju uopštenje pojma enumeracije, jer su tipovi čiji skupovi instanci ne mogu da se menjaju tokom izvršavanja programa, ali su, za razliku od enumeracija čije su instance proste, skalarne simboličke vrednosti, instance ovakvih klasa objekti sa svim svojim svojstvima
- Na primer, želimo da napravimo predefinisan, prekonfigurisan skup korisničkih uloga u programu koje se inicijalizuju statički, u toku prevođenja, pri čemu su te uloge predstavljene objektima koji imaju odgovarajuće usluge (npr. proveru da li korisnik sa tom ulogom može da izvrši datu komandu):

```
class UserRole {
public:
  constexpr UserRole (const char* name, …);
  const char* name = nullptr;

  bool isAuthorizedFor (Command* cmd) const;
  …
};

constexpr UserRole ordinary("Ordinary user", …);
constexpr UserRole privileged("Privileged user", …);
constexpr UserRole admin("Administrator", …);
```

Konstruktor je *constexpr*, pa se može koristiti za konstantnu inicijalizaciju statičkih objekata u toku prevođenja, pod uslovom da se poziva sa argumentima koji su konstantni izrazi

Ovi objekti se inicijalizuju statički, konstantnom inicijalizacijom i konstantni su

Konstantna inicijalizacija

- Primer: struktura za alokaciju prostora za objekte tipa *T*, bez fragmentacije (ostataka slobodnog prostora):

```
template <class T, int size>
class Storage {
public:

  constexpr Storage () : head(slots) { slots[size-1].next = nullptr; }

  void* alloc () { Slot* p=head; if (p) head=p->next; return p->slot; }
  void  free (void* addr) { head = new (addr) Slot(head); }

private:

  struct Slot {

    constexpr Slot () : next(this+1) {}
    Slot (Slot* nxt) : next(nxt) {}

    union {
      Slot* next;
      char slot[sizeof(T)];
    };

  };

  Slot* head;
  Slot slots[size];

};

Storage<Clock,50> clockStorage;
```