Nabrajanja (enumeracije)

- Enumeracije *nisu* fundamentalan, celobrojan tip (nisu isto što i *int*), iako postoji implicitna konverzija iz tipa enumeracije u celobrojni tip
- Enumeracije treba koristiti kad god postoji potreba za tipom podataka koji apstrahuje koncept iz domena problema (ili rešenja) i čije instance uzimaju vrednosti iz konačnog, diskretnog skupa vrednosti koje se simbolički imenuju
- Nije dobro koristiti fundamentalan, celobrojni tip umesto enumeracije u ovakvim slučajevima, zato što:
  - je program tada teži za razumevanje, jer je teško razumeti značenje celobrojnih konstanti i nije ih lako razlikovati od drugog značenja istih konstanti (npr. kao zaista instanci matematičkog koncepta celog broja)
  - za vrednosti apstraktnog tipa enumeracije po pravilu nisu definisane (konceptualno) aritmetičke operacije (sabiranje, oduzimanje, množenje, deljenje itd)
  - program je manje fleksibilan: ako je potrebno promeniti odluku o implementaciji apstraktnog tipa, ili promeniti preslikavanje na celobrojne vrednosti, potrebno je zameniti sve pojave tih celobrojnih konstanti i pažljivo ih razlikovati od upotrebe istih vrednosti sa potpuno različitim značenjem
- Neke tipične upotrebe:
  - statusi operacija (funkcija)
  - stanja objekata
  - komande, signali i slično
  - statusi ili komande hardverskih uređaja
  - tip sa malim, konačnim i konstantnim, predefinisanim skupom instanci (npr. dani u nedelji, meseci u godini)

Nabrajanja (enumeracije)

- Postoji implicitna konverzija iz tipa enumeracije u celobrojni tip, koja spada u skup celobrojnih promocija. Vrednost tipa enumeracije konvertuje se u:
  - prvi od sledećih tipova koji može da prihvati ceo opseg vrednosti te enumeracije, ukoliko nije eksplicitno zadat potporni tip: *int*, *unsigned int*, *long*, *unsigned long*, *long long*, or *unsigned long long*
  - u potporni tip enumeracije, kao i u tip u koji se on može promovisati, ako je potporni tip eksplicitno zadat
- Međutim, takva konverzija najčešće nije adekvatna, jer meša tipove i suprotna je navedenim principima. Samo u posebnim slučajevima koje treba dobro lokalizovati i enkapsulirati, ovo ima smisla. Na primer:

```
enum Status { initiated, suspended, committed, canceled, failed };
int i = suspended; // i gets the value 1
Status s = canceled;
…

if (s==4) ... // conversion Status->int evaluates to true if s==failed
```