Obrada izuzetaka

- Preporučeni stil tretiranja izuzetaka jeste taj da se, prema ovoj paradigmi, izuzetkom tretira nemogućnost zadovoljenja nekog od ovih logičkih uslova, i *samo* to (sve drugo nisu izuzeci):
  - preduslovi (*precondition*): pozvani potprogram, ukoliko nije ispunjen njegov preduslov (npr. neregularan argument) treba da podigne izuzetak; na primer, neispravan parametar
  - postuslovi (*postcondition*): ukoliko potprogram nije u mogućnosti da ispuni svoje postuslove, treba da signalizira izuzetak svom pozivaocu; na primer, funkcija ne može da kreira povratnu vrednost ili ne može da uspostavi njenu invarijantu, ne može da pronađe ono što bi morala da pronađe i vrati, i slično
  - invarijante (*invariant*): ukoliko metoda klase ne može da očuva invarijantu, treba da baci izuzetak; invarijante su zapravo preduslovi i postuslovi koji važe za objekat na ulazu i izlazu svake metode klase
- Jedna tehnika za apstrahovano ispitivanje uslova - tzv. *tvrdnje* (*assertion*):

```
assert(i>=0 && i<this->size);
```

- U standardnoj biblioteci za C++, *assert* je definisan kao makro koji zavisi od drugog definisanog makroa *NDEBUG* koji nije definisan u biblioteci, već se može definisati (uključiti ili isključiti) u okruženju prevodioca:
  - ako je *NDEBUG* definisan (uključen), makro *assert* nema nikavog efekta (zamenjuje se sa `((void)0)`)
  - u suprotnom, zamenjuje se implementacijom koja ispituje vrednost datog skalarnog izraza; ako je ta vrednost nula, na standardni izlaz za greške ispisuje informacije o mestu u programu i završava program
- Ovakvi slični makroi ili funkcije mogu da se koriste za ispitivanje preduslova, postuslova i invarijanti, ali tako da podižu izuzetke ukoliko uslov nije ispunjen

Obrada izuzetaka

- Tretman izuzetaka, odnosno izuzetnih situacija koje su detektovane, može da bude na jednom od sledećih nivoa sigurnosti (*safety*), od najstrožijeg ka najlabavijem:
  - *Nothrow* (ili *nofail*) garancija: potprogram nikada ne baca izuzetak (funkcije označene kao *noexcept*); to znači da nema preduslove i uvek će sigurno očuvati sve invarijante i ispuniti svoje postuslove, ili pak greške sakriva ili tretira na drugi način; ovakvo ponašanje očekuje se od destruktora (oni su implicitno *noexcept*) i *move* konstruktora i operatora dodele, kao i drugih funkcija koje se implicitno pozivaju tokom prosleđivanja bačenog izuzetka do hvatača
  - *Jaka garancija sigurnosti od izuzetaka* (*strong exception safety guarantee*): ako potprogram baci izuzetak, stanje programa biće vraćeno na ono pre poziva tog potprograma (tzv. *rollback*) - sam potprogram je napravljen tako da to stanje povrati ili očuva
  - *Osnovna garancija sigurnosti od izuzetaka* (*basic exception safety guarantee*): ako potprogram baci izuzetak, stanje programa biće validno (iako može biti različito od onog pre poziva tog potprograma), tj. sve invarijante biće očuvane
  - *Nikakva garancija*: ukoliko se baci izuzetak, program može ostati u nekorektnom stanju; ovakve pojave predstavljaju *bagove* (*bug*), odnosno nepravilnosti u programu koje treba rešavati
- Preporuka je da svaka funkcija podrži najstrožiji od ovih redom iznesenih nivoa koji je moguće ispuniti