Nestalni tipovi

- Kao *cv-kvalifikator* (*cv-qualifier*) objektnog tipa, potpuno analogno sa kvalifikatorom *const*, može da se koristi kvalifikator *volatile* (nestalan, nepostojan): označava da se vrednost ovog objekta može promeniti nezavisno od toka kontrole koda u kome se koristi, tipično od strane:
  - hardvera: u memoriju u kojoj je objekat uskladišten neki hardverski uređaj može *asinhrono*, tj. potpuno nezavisno od operacija programa, u proizvoljnim, nepredvidivim trenucima, upisati tj. promeniti njegovu vrednost
  - *signala*, odnosno rutina koje obrađuju asinhrone signale (*signal handler*) koji dolaze od operativnog sistema (ili hardvera, ali koje operativni sistem pretvara u asinhrone signale programu)
- Ovaj kvalifikator nalaže prevodiocu da ne vrši optimizacije koda i premeštanja operacija sa ovakvim objektom, što preciznije znači da sve operacije koje se po semantici izvršavaju pre neke date operacije čitanja ili promene ovakvog objekta moraju završiti pre te operacije, a one koje su po semantici iza te operacije ne smeju početi pre nego što se ta operacija završi
- Drugim rečima, prevodilac će svaku operaciju čitanja ili upisa u nestalan objekat izvršiti bez optimizacija i promena redosleda koda, ne sme je pročitati ili upisati pre nego što su sve operacije pre nje završene i svaki put je čita/upisuje iznova, iako program možda ne menja vrednost tog objekta
- Kvalifikator *volatile* se koristi potpuno analogno kao i kvalifikator *const*, i sva navedena pravila vezana za pokazivače, reference i konverzije važe na potpuno isti način
- Objekat može biti istovremeno i *const* i *volatile*

Korisnički definisane konverzije

- Konstruktor klase *X* koji se može pozvati samo sa jednim stvarnim argumentom tipa *T* definiše *korisničku konverziju* (*user-defined conversion*) iz tipa *T* u tip *X* i naziva se *konverzioni konstruktor* (*converting constructor*):

```
class X {
public:
  X (int);
};
```

- Funkcija se može pozvati sa samo jednim stvarnim argumentom tipa *T* ako ima samo jedan formalan parametar, ili ako svi drugi formalni parametri imaju podrazumevane vrednosti; taj prvi (ili jedini) parametar mora biti tipa *T* ili tipa reference na *T* (obično tipa *const T&*)
- Ova konverzija može se vršiti eksplicitno, bilo kojim operatorom konverzije, ali i implicitno, gde god se vrši implicitna konverzija; u nizu tranzitivnih implicitnih konverzija koje se mogu vršiti od datog do odredišnog tipa, samo jedna može biti korisnička konverzija:

```
X f (X x1) {
  …
  return 2;
}

void g () {
  X x2 = f(1);
}
```