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);
};
```

Konstruktor konverzije iz tipa *int* u tip *X*

- 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);
}
```

Prilikom povratka iz funkcije *f*, povratna vrednosti tipa *X* dobija se konverzijom izraza iza naredbe *return* koji je tipa *int*: vrši se implicitna konverzija iz *int* u *X* pozivom konverzionog konstruktora: *X x2(2)*

Prilikom poziva funkcije *f(1)*, formalni argument *x1* tipa *X* inicijalizuje se stvarnim argumentom tipa *int*: vrši se implicitna konverzija iz *int* u *X* pozivom konverzionog konstruktora: *X x1(2)*

Korisnički definisane konverzije

- Konverzionim konstruktorom klase *X* može se definisati korisnička konverzija iz bilo kog drugog tipa, pa i ugrađenog (neklasnog) tipa, u tip *X*, ali ne i obrnuto, iz tipa *X* u neki ugrađeni tip
- Takva konverzija može se definisati *operatorom konverzije*, kao nestatičkom operatorskom funkcijom članicom klase *X*, koja nema parametre, nema eksplicitan povratni tip, i ima ime u kome se koristi deklarator tipa (*type-id*); ovakva funkcija treba da vrati vrednost datog tipa:

```
class X {
public:
  operator int ();
  operator Y* ();
};
```

- Kao *type-id* može se navesti bilo koji fundamentalni ili složeni tip, ali se ne mogu upotrebljavati simboli za niz *[]* i funkciju *()*, osim indirektno, kroz *typedef* sinonime, s tim da odredišni tip ne može biti niz ili funkcija čak ni tako
- I ova konverzija može se vršiti eksplicitno, bilo kojim operatorom konverzije, ali i implicitno, gde god se vrši implicitna konverzija; na primer:

```
X x;
int i = x;
Y* py = x;
```