Tipovi

- Za svaki od tipova, osim tipova funkcija ili referenci, postoje tzv. *cv-kvalifikovane* (*cv-qualified*) varijacije: *const* (konstantni), *volatile* (nestalni) i *const volatile*
- Tipovi su grupisani i u različite kategorije po nekim svojim svojstvima:
  - *objektni tipovi* (*object types*) su (potencijano cv-kvalifikovani) tipovi koji nisu tipovi funkcija, referenci i tip *void*; objekti mogu biti nekog od ovih tipova
  - *skalarni tipovi* (*scalar types*) su (potencijano cv-kvalifikovani) tipovi koji nisu tipovi nizova ili klasni tipovi; entiteti ovih tipova nemaju u sebi ugrađene objekte
    - *trivijalni tipovi* (*trivial types*), tzv. *POD* (*Plain Old Data*) tipovi, tipovi literala (*literal types*) i drugi
- Za svaku od navedenih kategorija i potkategorija, u standardnoj biblioteci postoji odgovarajuća šablonska struktura *is_…* sa podatkom članom *value* koji u vreme prevođenja rezultuje vrednostima *true* ili *false*, u zavosnosti od toga da li je tip koji je argument šablona pripadnik date kategorije; na primer:

```
bool b = std::is_integral<float>::value; // b==false
b = std::is_reference<float&>::value; // b==true
```

- Specifikator *decltype* određuje tip datog entiteta ili tipa izraza datog u zagradama, na osnovu deklaracija, i to u vreme prevođenja (izraz kao operand se ne izračunava za vreme izvršavanja, već se njegov tip određuje za vreme prevođenja); ovo se može koristiti za definisanje tipova koje je teško odrediti, ili koji zavise od tipova drugih entiteta, tipično u šablonima; na primer:

```
b = is_reference<decltype(x+y)>::value;
```

Biće *true* akko operator + nad operandima *x* i *y* vraća tip reference

Tipovi

- Neko ime (identifikator) se može deklarisati kao ime tipa na jedan od sledećih načina:
  - deklaracijom klase
  - deklaracijom enumeracije (*enum*)
  - *typedef* deklaracijom
  - deklaracijom sinonima za definisani tip
- *typedef* deklaracija ima isti oblik kao kada se imenom deklariše entitet datog tipa, samo što ime ne deklariše takav entitet, već novo ime (sinonim) za taj isti tip (ne definiše se novi tip); tipično se koristi za konciznije pisanje složenih tipova:

```
typedef X* (*PXTransFun)(X*);
PXTransFun vtable[N];
```

- Potpuno isto se postiže sledećom deklaracijom sinonima za tip:

```
using PXTransFun = X*(*)(X*);
```

- Na mnogim mestima može se navesti neki tip koji nema ime, na primer kao odredišni tip operatora eksplicitne konverije (*cast*); takav bezimeni tip piše se isto kao i tip u deklaraciji *typedef*, samo bez imena, odnosno kao u deklaraciji sinonima i naziva se *type-id*