Prostori imena

- Prostori imena se mogu proizvoljno ugnežđivati; kvalifikovani pristup imenu iz ugnežđene oblasti važenja vrši se višestrukim nadovezivanjem operatora ::, npr. *domain::core::users::User*
- U svakoj jedinici prevođenja podrazumeva se jedan implicitno definisan, bezimeni prostor imena, tzv. *globalni opseg važenja* (*global scope* ili *file scope*); ime deklarisano van svih definicija funkcija, klasa ili prostora imena ima *globalnu oblast važenja* ovog implicitnog prostora imena, počev od tačke deklarisanja, do kraja fajla u kom je ta deklaracija
- U nekoj ugnežđenoj oblasti važenja u kojoj je globalno ime skriveno, tom imenu se može pristupiti kvalifikacijom preko unarnog operatora :: (bez levog operanda):

```
int x = 0;
void f () {
  int x = 2;
  x = 3;
  ::x = 3;
}
int* p = &x;
```

Globalno ime *x*, važi od tačke deklarisanja, do kraja ovog fajla

Lokalno ime *x*, sakriva globalno ime *x*

Odnosi se na lokalno *x*

Odnosi se na globalno *x*

Odnosi se na globalno *x*, jer je lokalna oblast važenja bloka završena

- Može se definisati i bezimen prostor imena (*unnamed namespace*): opseg važenja imena deklarisanih u njemu uključuje i okružujuću oblast važenja, ali takva imena imaju interno vezivanje; na ovaj način se mogu deklarisati globalna imena sa internim vezivanjem (umesto stare upotrebe specifikatora *static*), odnosno delovi implementacije nekog modula koji su skriveni od drugih modula:

```
namespace {
  ...
}
```

Enumeracija sa opsegom važenja

- Enumeracije koje su do sada pominjane uvode enumeratore (simboličke konstante) u okružujući opseg važenja, što znači da se imena tih enumeratora mogu sukobiti sa drugim istim imenima u oblasti važenja u kojoj je ta enumeracija definisana
- Da bi se ovo sprečilo, enumeracija se može deklarisati tako da bude *sa opsegom važenja* (*scoped enumeration*), navođenjem ključne reči *class* ili *struct*, svejedno:

```
enum class Color { red, green, blue };
```

- Sada su enumeratori u oblasti važenja svoje enumeracije, pa se van te oblasti važenja mogu koristiti samo kvalifikovano:

```
Color r = Color::blue;

switch (r) {
    case Color::red  : std::cout << "red\n";   break;
    case Color::green: std::cout << "green\n"; break;
    case Color::blue : std::cout << "blue\n";  break;
}
```

- Za ovakve enumeracije ne postoji implicitna konverzija u integralne tipove; može se vršiti eksplicitna konverzija operatorom *static_cast*