Opseg važenja bloka

- Opštije, varijable se mogu deklarisati na sledećim mestima unutar naredbi, i tada takva imena imaju opseg važenja samo do kraja te naredbe:
  - kao inicijalna naredba naredbi *if, switch* i *for*:

```
const string myString = "My Hello World Greeting";
if (const auto it = myString.find("Hello"); it != string::npos)
    cout << it << " Hello\n";
if (const auto it = myString.find("World"); it != string::npos)
    cout << it << " World\n";
```

Ovo je inicijalna naredba naredbe *if*

Ovo je uslov naredbe *if*

Opseg važenja imena *it* je samo naredba *if*, pa se iza kraja te naredbe može ponovo definisati nova varijabla sa istim imenom

  - kao uslov naredbi *if, switch, while* i *for*:

```
Base* pb = …;
if (Derived* pd = dynamic_cast<Derived*>(pb)) {
    pd->f();
    …
}
```

Opseg važenja imena *pd* je samo naredba *if*, odnosno blokovi njenih *then* i *else* delova, pa se iza kraja te naredbe može ponovo definisati nova varijabla sa istim imenom

Opseg važenja bloka

- I klasa može biti definisana unutar tela funkcije - tzv. *lokalna klasa* (*local class*). Takva klasa nije dostupna van opsega važenja te funkcije i ima neka ograničenja: ne može imati statičke podatke članove, njene funkcije članice moraju biti definisane unutar definicije te klase i drugo
- Ovakve klase tipično se prave da bi se na licu mesta napravili ad-hoc objekti koji zadovoljavaju neki interfejs i samo redefinišu neke polimorfne operacije, tj. objekti su neke klase izvedene iz neke apstraktne klase, s tim da su takve pojave bitne samo na datom mestu, a ne šire, odnosno takvi objekti se prave samo tu, dok ih svi ostali vide samo kroz te apstraktne, generalizovane interfejse, a nikad kao instance te lokalne klase (pa je zato ta klasa i lokalna, enkapsulirana u funkciju)
- Na primer, neka funkcija očekuje kao parametar objekat koji zadovoljava neki interfejs, kako bi pozvala neku njegovu operaciju (tzv. *callback* mehanizam):

```
class ISubscriber {
public:
  virtual void notify (Message*);
};

class Publisher {
public:
  static void subscribe (ISubscriber*);
};

void Component::Component () {
  struct Sub : public ISubscriber {
    Sub (Component* c) : comp(c) {}
    virtual void notify (Message* msg) { if (comp) comp->processMessage(msg); }
    Component* comp;
  };

  Publisher::subscribe(new Sub(this));
}
```