Opseg važenja bloka

- Inicijalna naredba naredbe *for* može biti deklaracija varijable; njeno ime ima potencijalni opseg važenja od tačke deklarisanja do kraja naredbe koja čini telo te naredbe *for*, i ne važi van te naredbe. Na primer:

```
void read (int a[], int b[], int size) {

  for (int i=0; i<size; i++)
    cin>>a[i];

  for (int i=0; i<size; i++)
    cin>>b[i];

}
```

Opseg važenja imena *i* je samo naredba *for*, pa se iza kraja te naredbe može ponovo definisati nova varijabla sa istim imenom, pošto prethodna više ne važi

- Međutim:

```
template<typename T>
int search (T array[], int size, T x) {

  for (int i=0; i<size && array[i]!=x; i++);

  if (i<size) return i;
  else return -1;

}
```

Opseg važenja imena *i* je samo naredba *for*

Greška u prevođenju, *i* nije u opsegu važenja

- Ako je potrebno koristiti ovu varijablu i nakon petlje, potrebno je deklaraciju izvući izvan naredbe *for*:

```
template<typename T>
int search (T array[], int size, T x) {

  int i=0;
  for (; i<size && array[i]!=x; i++);

  if (i<size) return i;
  else return -1;

}
```

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";
```

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

```
Base* pb = …;

if (Derived* pd = dynamic_cast<Derived*>(pb)) {

    pd->f();
    …

}
```