

# C++-Prostor imena

Aleksandra Trailović  
aleksandratrailovic97@gmail.com

15. decembar 2018.

## 1 Uvod

[4] C++ uključuje specifikaciju, *prostor imena* (eng. *namespace*), koji pomaže pri upravljanju problemom globalnog zagađivanja imena.[5] Do zagađivanja dolazi pri upotrebi velikog broja različitih biblioteka i pri pisanju velikih programa. Zagađivanje ima više neugodnih posledica: isto ime može da se upotrebljava u više biblioteka, ili greškom može da se navede pogrešno ime koje postoji u nekoj drugoj biblioteci.

[2] *Prostor imena* je dodatak u C++ i nije prisutan u C-u. To je deklarativna regija pruža opseg identifikatora (imena tipova, funkcija, promenljivih) unutar nje. Dozvoljeno je više blokova prostora imena sa istim imenom.

[1] Definicija prostora imena počinje ključnom rečju `namespace` i nazivom imenskog prostora:

```
namespace indetifier
{
    named_entities
}
```

gde je `indetifier` bilo koji validni identifikator i `named entities` skup promenljivih tipova i funkcija koje su uključene unutar prostora imena. Na primer:

```
namespace myNamespace
{
    int a, b;
}
```

gde su promenljive `a` i `b` normalne promenljive deklarisanе u imenskom prostoru `myNamespace`.

Njima se može pristupiti iz njihovog imenskog prostora, sa njihovim identifikatorima, ali, ako se pristupi van `myNamespace` prostora imena, moraju biti

pravilno kvalifikovane operatorom opsega ::. Na primer, da bismo pristupili prethodnim promenljivama van myNamespace, one moraju biti kvalifikovane kao:

```
myNamespace::a  
myNamespace::b
```

**Primer 1.1**

|  |   |
|--|---|
| <pre>// namespaces<br/>#include &lt;iostream&gt;<br/>using namespace std;<br/><br/>namespace foo<br/>{<br/>    int value() { return 5; }<br/>}<br/><br/>namespace bar<br/>{<br/>    const double pi = 3.1416;<br/>    double value() { return 2*pi; }<br/>}<br/><br/>int main () {<br/>    cout &lt;&lt; foo::value() &lt;&lt; '\n';<br/>    cout &lt;&lt; bar::value() &lt;&lt; '\n';<br/>    cout &lt;&lt; bar::pi &lt;&lt; '\n';<br/><br/>    return 0;<br/>}</pre> | <pre>outer block:<br/>5<br/>6.2832<br/>3.1416</pre> |
|--|---|

Slika 1: U ovom slučaju, postoje dve funkcije sa istim name:value. Jedna je definisana u okviru prostora imena foo, a druga u okviru prostora imena bar.

## 2 STD prostor imena

[1] Sve promenljive, tipovi, konstante i funkcije standardne C++ biblioteke deklarirane su unutar std prostora imena. Naime, većina primera koje smo videli uključuje sledeću liniju: *using namespace std;* Ovo uvodi direktnu vidljivost svih imena std prostora imena u kodu. Uvodi se kako bi se olakšalo razumevanje i skratila dužina zapisa.

Na primer, umesto: `std::cout << "Hello world!";` možemo pisati: `cout << "Hello world!";`

### 3 USING

[1] Ključna reč *using* uvodi ime u trenutnu deklarativnu regiju(kao što je blok), čime se izbegava potreba za kvalifikacijom imena.

Primer 3.1

|  |  |
|--|--|
| <pre>// using #include &lt;iostream&gt; using namespace std;  namespace prvi {     int x = 5;     int y = 10; }  namespace drugi {     double x = 3.1416;     double y = 2.7183; }  int main () {     using prvi::x;     using drugi::y;     cout &lt;&lt; x &lt;&lt; '\n';    //5     cout &lt;&lt; y &lt;&lt; '\n';    //2.7183     cout &lt;&lt; prvi::y &lt;&lt; '\n'; //10     cout &lt;&lt; drugi::x &lt;&lt; '\n'; //3,1416     return 0; }</pre> | <pre>outer block: 5 2.7183 10 3.1416</pre> |
|--|--|

Slika 2: Primitimo kako u funkciji main, promenljiva(bez kvalifikatora imena) referiše na prvi::x, dok promenljiva y referiše na drugi::y, baš kao što je navedeno u deklaracijama koje koriste.

### 4 Klase i prostor imena

[2] Klasa se može definisati u prostoru imena:

Primer 4.1

```
#include <iostream>
using namespace std;

namespace ns
```

```

{
    class geek; //samo deklariseмо klasu
}

class ns::geek //definisanje klase spolja
{
    public:
        void display()
        {
            cout<<"ns::geek::display()\n";
        }
};

int main()
{ //Kreiranje objekta klase student
    ns::geek obj;
    obj.display();
    return 0;
}

```

---

Klasa se takođe može definisati unutar prostora imena i izvan prostora imena.

#### Primer 4.2 C++ program pokazuje korišćenje klase u prostoru imena

---

```

#include <iostream>
using namespace std;

namespace ns
{
    class geek //klasa u prostoru imena
    {
    public:
        void display()
        {
            cout<<"ns::geek::display()\n";
        }
    };
};

int main()
{ //Kreiranje objekta klase student
    ns::geek obj;
    obj.display();
    return 0;
}

```

---

Metode možemo deklarirati i izvan prostora imena.

#### Primer 4.3 C++ program pokazuje definisanje metoda izvan prostora imena

---

```

#include <iostream>
using namespace std;

```

```

namespace ns
{
    void display();
    class geek
    {
    public:
        void display();
    };
}

void ns::geek::display() // Definisiranje metoda prostora imena
{
    cout << "ns::geek::display()\n";
}
void ns::display()
{
    cout << "ns::display()\n";
}

int main()
{
    ns::geek obj;
    ns::display();
    obj.display();
    return 0;
}

```

---

[2]

## 5 Proširenje prostora imena i neimenovani prostor imena

[3] Moguće je kreirati više od jednog prostora imena u globalnom prostoru. Postoje dva načina:

- Dajući prostorima imena različita imena
- Proširenjem prostora imena (Koristeći isto ime više puta)

### 5.1 Prostori imena imaju različita imena

Sledeći C++ program pokazuje više od jednog imenskog prostora sa različitim imenima:

---

```

#include <iostream>
using namespace std;

namespace prvi
{
    int f(){ return 5; }
}

```

```

namespace drugi
{
    int f(){ return 10; }
}

int main()
{
    cout<<prvi::f()<<"\n";
    cout<<drugi::f()<<"\n";
    return 0;
}

```

---

### Output

---

```

5
10

```

---

## 5.2 Proširenje prostora imena

Sledeći C++ program pokazuje proširenje prostora imena

---

```

#include <iostream>
using namespace std;

namespace prvi //prvi imenski prostor
{
    int vrednost1=500;
}

namespace prvi //ostali deo prvog imenskog prostora
{
    int vrednost2=501;
}

int main()
{
    cout <<prvi::vrednost1 <<"\n";
    cout <<drugi::vrednost2 <<"\n";
    return 0;
}

```

---

### Output

---

```

500
501

```

---

## 5.3 Neimenovani prostori imena

U neimenovanim prostorima imena, ime prostora imena nije navedeno u deklaraciji prostora imena. Sledeći C++ program pokazuje neimenovani prostor

imena

---

```
#include <iostream>
using namespace std;

namespace
{
    int rel=300;
}

int main()
{
    cout << rel << "\n";
    return 0;
}
```

---

## Literatura

- [1] cplusplus.com. Namespaces. on-line at: <http://www.cplusplus.com/doc/tutorial/namespaces/>.
- [2] GeeksforGeeks. Namespace in C++. on-line at: <https://www.geeksforgeeks.org/namespace-in-c/>.
- [3] GeeksforGeeks. namespace in C++ | Set 2 (Extending namespace and Unnamed namespace). on-line at: <https://www.geeksforgeeks.org/namespace-in-c-set-2-extending-namespace-and-unnamed-namespace/>.
- [4] R. Sebesta. *Concepts of programming languages*. 2012.
- [5] Milena Vujošević-Janičić. Dizajn programskih jezika, 2018. on-line at: [http://www.programskijezici.matf.bg.ac.rs/dpj/2017/predavanja/04/polimorfizamCplusplus\\_tekst.pdf](http://www.programskijezici.matf.bg.ac.rs/dpj/2017/predavanja/04/polimorfizamCplusplus_tekst.pdf).