Prof. Dr. Agni Dika
Material punues.
Nuk është për shumëzim!

0 3

Hapësira e emrave

Hyrje

Në librari të ndryshme mund të përdoren edhe *identifikator global*. Por, ndodhë që identifikator të njëjtë të shfrytëzohet në disa librari. Nëse libraritë e tilla përdoren njëkohësisht në një program, kompjuteri nuk do të jetë në gjendje t'i dalloj identifikatorët. Kështu, p.sh. ne mund të gabojmë gjatë definimit të një klase, duke e quajtur ate me emrin e një klase që është definuar më parë brenda një librarije të cilën jemi duke e shfrytëzuar.

Me qëllim të kufizimit të dukshmërisë së elementeve të programeve (konstanteve me emra, variablave ose funkcioneve), ato deklarohen brenda strukturave, klasave ose fajllave. Por, një gjë e tillë është më rëndë të realizohet, p.sh. për emrat e strukturave, klasave ose objekteve. Për këtë qëllim mund të përdoren *hapësira emrash* (angl. namespace), përmes së cilave *dukshmëria e emrave kufizohet vetëm në hapësirën ku ato definohen*.

Në programet standarde zakonisht shfrytëzohen *hapësira standarde e emrave* (`std`). Për këtë qëllim, në fillim të programit vendoset direktiva:

```
using namespace std;
```

Brenda kësaj hapësire standarde p.sh. përfshihen emrat **`cin`**, **`cout`** etj. Por, nëse në fillim të programit nuk shënohet direktiva në fjalë, është i pamundur përdorimi i këtyre komandave, përkatësisht kompjuteri do të gjeneroj mesazhe për gabim.

Shembull

```
// Programi prg1
#include <iostream>
int main()
{
    int a;
    cout << "\nVlera a: ";
    cin >> a;
    cout << "\nVlera e lexuar a="
         << a
         << "\n\n";
     return 0;
}
```

Nëse ekzekutohet programi në formën e dhënë, për komandat **`cin`** dhe **`cout`** kompjuteri do të na njoftojë se ato janë `undeclared identifier`.

Nënkuptohet se gabimet në fjalë do të eliminohen nëse shfrytëzohet direktiva:

```
using namespace std;
```

ose nëse para komandave në fjalë shkruhet emri i hapësirës **std**, në të cilën ato janë definuar si dhe operatori katër pika (**::**), kështu:

```
// Programi prg2
#include <iostream>
int main()
{
     int a;
     std::cout << "\nVlera a: ";
     std::cin >> a;
     std::cout << "\nVlera e lexuar a="
          << a
          << "\n\n";
      return 0;
}
```

Definimi i hapësirave me emra

Hapësira me emra, përveç nga shtëpitë e ndryshme softverike, mund të definohen edhe nga të gjithë ata që shkruajnë programe. Për definim të një hapësire me emra shfrytëzohet fjala **namespace**, e cila pastaj pasohet me emrin e hapësirës që definohet.

Shembull

Definimi i hapësirës me emra **jeta**.

```
namespace jeta
{
     int i,m=5;
     void dita()
     {
          for (i=1;i<=m;i++)
              cout << "\nDita e bukur";
     }
}
```

Përmes kësaj pjese të programit definohet *hapësira e emrave* **jeta**, në të cilën përfshihen identifikatorët e variablave **i** e **m** dhe funkcionit **dita()**, të cilët njihen edhe si *anëtarë të hapësirës së emrave*.

Shembull

Programi në të cilin tregohet qasja te variabla **m** dhe funkcioni **dita()**, të cilët janë përfshirë në hapësirën e emrave **jeta**.

```
// Programi prg3
#include <iostream>
using namespace std;
namespace jeta
{
     int i,m=5;
     void dita()
     {
          for (i=1;i<=m;i++)
              cout << "\nDita e bukur";
     }
```

```
}

int main()
{
     cout << "\nVlera e variables m="
          << jeta::m
          << "\n";
     jeta::dita();
     cout << "\n\n";
return 0;
}
```

Siç shihet për qasje këtu është shfrytëzuar operatori katër pika.

Nëse paraprakisht hapësira e emrave **jeta** vendoset në disk, si fajll tekstual me emrin **omega.txt**, programi do të duket kështu:

```
// Programi prg4
#include <iostream>
using namespace std;
#include "C:\Alfa\omega.txt"
int main()
{
     cout << "\nVlera e variables m="
          << jeta::m
          << "\n";
     jeta::dita();
     cout << "\n\n";
return 0;
}
```

Në program nuk do të duhej të shfrytëzohej *operatori katër pika* gjatë qasjes te variabla **m** dhe funkcioni **dita**, nëse paraprakisht shfrytëzohet direktiva:

```
using namespace jeta;
```

ashtu siç shihet në vijim.

```
// Programi prg5
#include <iostream>
using namespace std;
#include "C:\Alfa\omega.txt"
using namespace jeta;
int main()
{
     cout << "\nVlera e variables m="
          << m
          << "\n";
     dita();
     cout << "\n\n";
return 0;
}
```

Direktiva:

```
using namespace jeta;
```

mund të vendoset kudo në program, por patjetër pas definimit të hapësirës me emra **jeta** dhe para shfrytëzimit të identifikatorëve që paraqiten brenda saj.

Qasja mund të relaizohet edhe duke e shfrytëzuar direktivën **using** për secilin identifikator, ashtu siç shihet në programin vijues.

```
// Programi prg6
#include <iostream>
using namespace std;
#include "C:\Alfa\omega.txt"
int main()
{
    using jeta::m;
    using jeta::dita;
    cout << "\nVlera e variables m="
         << m
         << "\n";
    dita();
    cout << "\n\n";
return 0;
}
```

Vlerat e variablave që përfshihen brenda hapësirës me emra mundet edhe të lexohen. Gjatë shkruarjes së komandës përkatëse për lexim veprohet plotësisht njëlloj si edhe gjatë shtypjes së vlerave të tyre.

Shembull

Verzioni i programit i cili u dha më sipër, në të cilin vlera e variablës m lexohet.

```
// Programi prg7
#include <iostream>
using namespace std;
namespace jeta
{
    int i,m;
    void dita()
    {
        for (i=1;i<=m;i++)
            cout << "\nDita e bukur";
    }
}

int main()
{
    cout << "\nVlera e variables m: ";
    cin >> jeta::m;
    jeta::dita();
    cout << "\n\n";
return 0;
}
```

Brenda hapësirës me emra mund të përfshihen edhe konstante.

Shembull

```
// Programi prg8
#include <iostream>
using namespace std;
namespace jeta
{
    double const pi=3.1415926;
    double perimetri(double r)
    {
        double p;
        p=2*pi*r;
        return p;
    }
}

int main()
{
    cout << "\nVlera e variablës pi="
        << jeta::pi
         << "\n";
    cout << "\nVlera e llogaritur p="
        << jeta::perimetri(5.5)
        << "\n\n";
return 0;
}
```

Definimi i njëkohshëm i disa hapësirave me emra

Në një program, përveç hapësirës standarde të emrave, mund të shfrytëzohen njëkohësisht edhe disa hapësira emrash, të definuara brenda programit ose të inseruara përmes direktivës paraprocesorike **#include**.

Shembull

Definimi i hapësirave me emrave **alfa** dhe **beta** në të cilat njëkohësisht paraqiten variabla **a** dhe **b**.

```
// Programi prg9
#include <iostream>
using namespace std;
namespace alfa
{
  int a = 10;
  double b = 55.55;
}
namespace beta
{
  double a = 77.77;
  int b = 20;
}
using namespace alfa;
using namespace beta;
int main ()
{
    cout << "\nVlera e variablës a: "
```

```
          << a << endl;
     cout << "\nVlera e variablës b: "
          << b << endl;
  return 0;
}
```

Në këtë rast për variablat **a** dhe **b** kompjuteri do ta lajmëroj gabimin `ambiguous symbol`.

Që të mos vijë deri te një gabim i tillë, kompjuterin duhet ta njohtojmë sakt se nga cila hapësirë me emra merren variablat, p.sh. si në programin vijues.

```
// Programi prg10
#include <iostream>
using namespace std;
namespace alfa
{
  int a = 10;
  double b = 55.55;
}
namespace beta
{
  double a = 77.77;
  int b = 20;
}
int main ()
{
     cout << "\nVlera e variablës a: "
          << alfa::a << endl;
     cout << "\nVlera e variablës b: "
          << beta::b << endl;
  return 0;
}
```

Në këtë rast do të shtypet vlera e variablës **a** që përfshihet në hapësirën **alfa**, kurse për variablën **b** do të shtypet vlera e përcaktuar në hapësirën **beta**.

Si zgjidhje më praktike këtu mund të merret ajo e shfrytëzimit të *direktivës* **using** vetëm për njërën hapësirë, p.sh. ashtu siç është dhënë në vijim.

```
// Programi prg11
#include <iostream>
using namespace std;
namespace alfa
{
  int a = 10;
  double b = 55.55;
}
namespace beta
{
  double a = 77.77;
  int b = 20;
}
using namespace alfa;
int main ()
{
```

```
    cout << "\nVlera e variablës a: "
         << a << endl;
    cout << "\nVlera e variablës b: "
         << beta::b << endl;
  return 0;
}
```

Problemi i konfliktit mund të paraqitet edhe nëse identifikatori i njëjtë paraqitet në një hapësirë dhe në programin kryesorë.

Shembull

```
// Programi prg12
#include <iostream>
using namespace std;
void dita();
namespace jeta
{
    int i;
    void dita()
    {
        for (i=1;i<=7;i++)
            cout << "\nDita e bukur";
    }
}

int main()
{
    jeta::dita();     // Prej namespace jeta
    dita();
    cout << "\n\n";
return 0;
}

void dita()
{
    int i;
    for (i=1;i<=5;i++)
        cout << "\nKoha me borë";
}
```

Këtu funkcioni me emrin **`dita()`** njëkohësisht definohet në programin kryesor dhe në hapësirën e emrave **`jeta`**.

Pjesa e pasistemuar

1.

```
// Programi prg13
#include <iostream>
using namespace std;
namespace alfa
{
      int a;
```

```
	double b;
}
using namespace alfa;
int main()
{
	int a,x;
	double y;
	alfa::a=5;
	a=4;
	alfa::b=7;
	x=3;
	y=a+2*x+alfa::a-2*alfa::b;   // b mund të shkruhet pa alfa
	cout << "\nRezultati y="
		<< y
		<< endl;
return 0;
}
```

2.

Vlerat jepen direkt.

```
// Programi prg14
#include <iostream>
using namespace std;

namespace alfa
{
	int a;
	double b=7;
}
using namespace alfa;
int main()
{
	int a=4,x;
	double y;
	alfa::a=5;
	x=3;
	y=a+2*x+alfa::a-2*alfa::b;
	cout << "\nRezultati y="
		<< y
		<< endl;
return 0;
}
```

3.

Definimi jasht i funkcionit që përfshihet në namespace.

```
// Programi prg15
#include <iostream>
using namespace std;

namespace alfa
{
	int a;
```

```
        const double b=7;
        void omega();
}
using namespace alfa;
int main()
{
        int a=4,x;
        double y;
        alfa::a=5;
        x=3;
        y=a+2*x+alfa::a-2*b;
        cout << "\nRezultati y="
                << y
                << endl;
        omega();            // Mund të shkruhat edhe alfa::omega()
return 0;
}

void alfa::omega()
{
        double z;
        z=a+b;                              // Shfrytëzohen variablat në alfa
        cout << "vlera z="
                << z
                << endl;
}
```

Nëse nuk përdoret:

```
        using namespace alfa;
```

patjetër duhet të shfrytëzohet operatori **::** për varibalat që janë pjesë e **alfa**.

```
// Programi prg16
#include <iostream>
using namespace std;

namespace alfa
{
        int a;
        const double b=7;
        void omega();
}
int main()
{
        int a=4,x;
        double y;
        alfa::a=5;
        x=3;
        y=a+2*x+alfa::a-2*alfa::b;
        cout << "\nRezultati y="
                << y
                << endl;
        alfa::omega();
return 0;
```

```
}

void alfa::omega()
{
      double z;
      z=a+b;
      cout << "vlera z="
           << z
           << endl;
}
```

Vazhdimi i deklarimit të hapësirës me emra

Hapësira me emra mund të vazhdohet të plotësohet me emra të rinjë.

Shembull

```
// Programi prg17
#include <iostream>
using namespace std;

namespace alfa
{
  int a = 10;
}

namespace beta
{
  double a = 77.77;
  int b = 20;
}

namespace alfa
{
  double b = 55.55;
}
int main () {
  using alfa::a;
  using beta::b;
  cout << "\nVersioni i parë:\n";
  cout << a << endl;
  cout << b << endl;
  cout << "\nVersioni i dytë:\n";
  cout << alfa::b << endl;
  cout << beta::a << endl;
  return 0;
}
```

Hapësira emrash të ndërthurura

Hapësira e emrave mund të deklarohet edhe brenda një hapësire tjetër.

Shembull

```
// Programi prg18
#include <iostream>
using namespace std;
namespace alfa
{
    int i,x=44,m=5;
    void dita()
    {
        for (i=1;i<=m;i++)
            cout << "\nDita e bukur";
    }
    namespace beta
    {
        double z=88.8;
    }
}

int main()
{
    cout << "\nVlera për x="
        << alfa::x
        << "\n";
    alfa::dita();    // Prej namespace të jashtëm
    cout << "\n\nVlera për z="
        << alfa::beta::z;
    cout << "\n\n";
return 0;
}
```

Shfrytëzimi lokal i hapësirës së emrave

Nëse direktiva **`using`** për hapësirë emrash shkruhet para programit, ajo ka efekt global dhe do të vlejë në komplet programin. Përndryshe, nëse diraktiva shkruhet brenda programit, vlefshmëria e saj do të kufizohet vetëm brenda bllokut që përcaktohet me kllapat e mëdha të hapura dhe mbyllura.

```
// Programi prg19
#include <iostream>
using namespace std;
namespace alfa
{
  int a = 77;
}

namespace beta
{
  double a = 55.44;
}

int main () {
  {
    using namespace alfa;
    cout << "\nVlera e hapësirës alfa: "
         << a
```

```
        << endl;
  }
  {
    using namespace beta;
    cout << "\nVlera e hapësirës beta: "
         << a
         << endl;
  }
  return 0;
}
```

Rez.

```
Vlera e hapësirës alfa: 77

Vlera e hapësirës beta: 55.44
```

Këtu shihet se direktivat e veçanta using kanë efekt vetëm brenda blloqeve të programit që përcaktohen me kllapat.

Nëse mes blloqeve të përcaktuara me dy kllapat e mëdha shkruhet komanda:

```
    cout << "\nVlera pas bllokut t% par%: "
         << a
         << endl;
```

kompjuteri do të lajmëroj:

```
error C2065:'a': undeclared identifier
```

Me këtë duhet nënkuptuar ate që u tha më sipër se efekti i direktivave **`using`** brenda programit kufizohet në blloqet që përcaktohen me kllapat e mëdha.

Emri alternativ i hapësirës së emrave

Hapësirës së emrave mund t'i shoqërohet edhe një emër alternativ, përkatësisht një pseudonim.

Shembull

```
// Programi prg20
#include <iostream>
using namespace std;
namespace jeta
{
  double x = 77.25;
}
namespace dita=jeta;  // Emri alternativ dita
using namespace dita;
int main()
{
```

```
    cout << "\nVlera n% hap%sir%n jeta: "
         << x
         << endl;
return 0;
}
```

Hapësirë emrash pa emër

Para programit lejohet deklarimi pa emër i hapësirës së emrave. Qasja te identifikatorët e kësaj hapësire nuk ka kufizim në asnjë pjesë të programit.

Shembull

```
// Programi prg21
#include <iostream>
using namespace std;
namespace
{
  double h = 33.66;
}
int main()
{
    cout << "\nVlera n% hap%sir%n pa em%r: "
         << h
         << endl;
return 0;
}
```

Funkcionet e mbingarkuara në hapësirën e emrave

Shembull

```
// Programi prg22
#include <iostream>
using namespace std;
namespace jeta
{
    double dita(int a)
    {
        return a;
    }
    int i;
    void dita()
    {
        for (i=1;i<=7;i++)
            cout << "\nDita e bukur";
    }
}
using namespace jeta;
int main()
{
```

```
    double y;
    dita();    // Prej namespace jeta
    y=dita(7);
    cout << y;
    cout << "\n\n";
return 0;
}
```
