Простори імен (Namespaces)

Потреба в просторах імен

```
1) int main() {
    int value;
    value = 0;
    double value; // Помилка
    value = 0.0;
}

2)
int value; // глобально десь в коді
...
int main() {
    double value; // Не помилка, але...
    value = 0.0;// Не помилка, але…
…
… value++; // це -локально...
}
```

```
3) //В файлі A:

int foo( int x) { return x; }

//В файлі B:

int foo( int x) { return x*2; }

//В файлі C:

#include “A”

#include “B”

…
z = foo(10); //??? ambiguity -
     двозначність
```

Namespace

```
namespace namespace_name {
   // декларації коду
 ***
}
#include <iostream>
  using namespace std;
  // Змінні, класи та функції даного
  // простору
  namespace first {
      int val = 500;
      void fun(void){
         cout<<”Make fun!”;
      }
  }
```

```
// Глобальні змінні
int val = 100;
void fun(void) {
        cout<<”nofun”;
}

int main() {
    // локальні змінні
    int val = 200;
    cout << first::val <<  val<<  '\n';
    fun();
    first::fun(); // виклик з first
    return 0;
}
```

Приклад

```
#include <iostream>
using namespace std;
// Перший name space
namespace first_space {
   void func() {
      cout << "Inside first_space" << endl;
   }
}
// Другий name space
namespace second_space {
   void func() {
      cout << "Inside second_space" << endl;
   }
}
int main () {
   // Функція з першого name space.
   first_space::func();
   // Функція з другого name space.
   second_space::func();
   return 0;
}
```

Keyword: using

```
// Всі метори та класи standard C++ визначені тут
namespace std {
…
/*All the files in the C++ standard library declare all of its entities within the std namespace. That is why we have generally included the using namespace std; statement in all programs that used any entity defined in iostream.
Всі файли STL в стандартному просторі імен (using namespace std), це визначено в iostream
*/

}
#include <iostream>
#include <string>
using namespace std; // Ми використовуємо стандартний namespace
****
string str1=”Hello”; // std::string
cout<<str1;  // std::cout<<str1;
```

```
//namespace std example
#include <iostream>
#include <cstdlib>

int main(){

std::cout<<"Demonstrating ";
using namespace std;
cout<<"the std namespace."<<endl;


}
```

```
//namespace std example
#include <iostream>
#include <cstdlib>

int main(){

std::string s1 = "Demonstrating ";

std::cout<<s1<<"the std
namespace."<<std::endl;

}
```

Використання using

```
#include <iostream>
using namespace std;
namespace first
{
  int x = 5;
  int y = 10;
}
namespace second{
  double x = 3.1416;
  double y = 2.7183;
}
int main () {
  using first::x;
  using second::y;
  cout << x << endl; //5
  cout << y << endl; //2.7183
  cout << first::y << endl;//10
  cout << second::x << endl;//3.14
  return 0;
}
```

```
#include <iostream>
using namespace std;
namespace first{
  int x = 5;
}
namespace second
{
  double x = 3.1416;
}
int main () {
  {
    using namespace first;
    cout << x << endl; // 5
  }
  {
    using namespace second;
    cout << x << endl;//3.14
  }
  return 0;
}
```

Приклад

```
#include <iostream>
using namespace std;
namespace ns {
  // Клас в просторі
class Student{
public:
   void display() {
        cout << "ns::Student::display()\n";
    }
 };
}
```

```
int main() {
// Створюємо об’єкт student Class
ns::Student obj;
 obj.display();
return 0; }
```

АБО:

```
using namespace ns;
int main() {
// Створюємо обєкт student Class
Student obj;
obj.display();
return 0; }
```

Директива та декларція using

```
#include <iostream>
#include <stdlib.h>
using namespace std;
class One{
        public:
        void funct1(char chs)
        {cout<<"character = "<<chs<<endl;}
};

class Two:public One{
        public:
        //using namespace One; /* директива using
не дозволена всередині класу */
        void funct1(char *str)
        {cout<<"String = "<<str<<endl;}
        //декларація using дозволена в класі
        using One::funct1;  //перевантаження
Two::funct1()
};
```

```
int main(){

   Two Sample;

   Sample.funct1('P'); //Виклик
One::funct1()


// Виклик Two::funct1()
   Sample.funct1(
                    const_cast<char*>
                       ("This is string"));
   return 0;
}
```

Приклад: визначення функцій ззовні

```
// C++ код, де визначено створює
функцію простора ззовні.
 #include <iostream>
using namespace std;
//Створюємо простір
namespace ns {

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

}
```

```
// Визначаємо метод
void ns::geek::display() {
   cout << "ns::geek::display()\n";
}
// другий метод
void ns::display() {
     cout << "ns::display()\n";
}
// Основна функція
int main() {
ns::geek obj;
ns::display();
obj.display();
 return 0;
}
```

Анонімні простора імен (Unnamed Namespaces):

- використовуються всередині програми для декларації унікальних ідентифікаторів unique identifiers;
- ім’я простору відсутнє в декларації простору – воно створюється компілятором;
- використвується лише всередині того файлу, де воно об’явлене;
- заміна статичних змінних та функцій.

```
// C++ програма з unnamed namespaces
#include <iostream>
using namespace std;
// декларація анонімного простору
namespace  {    int rel = 300;  }
int main() {
 cout << rel << "\n"; // prints 300
return 0;
}
```

Приклад

```
#include <vector>
namespace vec {
    template< typename T >
    class vector {
        // ...
    };

} // шаблон класу вектор

int main()
{
 // Стандартний вектор
   std::vector<int> v1;
 // наш вектор
    vec::vector<int> v2;
   // v1 = v2; // Error: v1 and v2
Об’єкти різних типів
 {
     using namespace std;
     vector<int> v3; //std::vector
     v1 = v3; // OK
 }
 {
     using vec::vector;
     vector<int> v4; //vec::vector
     v2 = v4; // OK
 }

  return 0;
}
```

Вкладені простори імен

```
// вкладені простори

#include <iostream>

using namespace std;

// вкладений namespace

namespace out {

    int val = 5;

    namespace in   {

        int val2 = val;

     }

}

 // Driver code

int main() {

cout << out::in::val2;

// prints 5

return 0; }
```

Приклад: визначення функцій ззовні

```
// C++ код, де визначено створює
функцію простора ззовні.
 #include <iostream>
using namespace std;
//Створюємо простір
namespace ns {

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

}
```

```
// Визначаємо метод
void ns::geek::display() {
   cout << "ns::geek::display()\n";
}
// другий метод
void ns::display() {
     cout << "ns::display()\n";
}
// Основна функція
int main() {
ns::geek obj;
ns::display();
obj.display();
 return 0;
}
```

Namespace Aliasing

```
//namespace new_name = current_name;

#include <iostream>
namespace name1  {
    namespace name2  {
        namespace name3  {
            int var = 42;
        }
    }
}
// Aliasing
namespace alias = name1::name2::name3;
int main() {
     std::cout << alias::var << '\n';
}
```