Пространство имен

Большинство приложений состоит более чем из одного исходного файла. При этом возникает вероятность дублирования имен, что препятствует сборке программы из частей. Для снятия проблемы в C++ был введен механизм логического разделения области глобальных имен программы, который получил название *пространства имен*.

Имена, определенные в пространстве имен, становятся локальными внутри него и могут использоваться независимо от имен, определенных в других пространствах. Таким образом, снимается требование уникальности имен программы.

namespace [<имя>] { <Объявления и определения> }

Имя пространства имен должно быть уникальным, но может быть и опущено. Если имя пространства опущено, то считается, что определено неименованное пространство имен, локальное внутри единицы трансляции. Для доступа к его ресурсам используется внутреннее имя $$$.

Например:

```
namespace ALPHA {              // ALPHA – имя пространства имен
      long double LD;          // объявление переменной
      float f(float y) { return y; }     // описание функции
      }
```

Доступ к элементам пространства имен

Пространство имен определяет область видимости, следовательно, функции, определенные в одном пространстве имен могут без ограничений использовать другие ресурсы, объявленные там же (переменные, типы и т.д.).

Доступ к элементам других пространств имен может осуществляться тремя способами:

1) с использованием квалификатора доступа, например:

```
ALPHA::LD или  ALPHA::f()
```

2) с использованием объявления using, которое указывает, что некоторое имя доступно в другом пространстве имен:

```
namespace BETA {          …
    using ALPHA::LD;/* имя ALPHA::LD доступно в BETA*/ }
```

3) с использованием *директивы using*, которая объявляет все имена одного пространства имен доступными в другом пространстве:

```
namespace BETA {          …
    using ALPHA;    /* все имена ALPHA доступны в BETA*/
  }
```

Непоименованное пространство имен

Непоименованное пространство имен невидимо в других модулях:

```
namespace { namespace-body }
```

При трансляции оно именуется как “unique”, доступное в самом модуле:

```
namespace unique { namespace-body }
using namespace unique;
```

Пример:

```
namespace { int i; }          // unique::i
void f() { i++; }             // unique::i++
namespace A {
    namespace { int i,j;}} // A::unique::i A::unique::j

using namespace A;
void h()
{   i++;       // unique::i или A::unique::i ?
    A::i++;    // A::i ?
    j++;      // A::unique::j++
}
```

Глобальное пространство имен

Приложение включает одно глобальное пространство имен. Имена, входящие в это пространство, объявляются без указания пространства имен.

Пример:

```
int i;

namespace A
{  int a, b, c;
   namespace B {int i, j, k;}
}

int main()
{
   A::a++;
   A::B::i++;
   ::i++;   // глобальное i
}
```

Имена стандартных библиотек C++

Согласно стандарту ANSI/ISO C++ все имена ресурсов стандартных библиотек определены в пространстве std. При этом подключаются файлы библиотек <cstdio>,<iostream>,<cmath> и т.д.

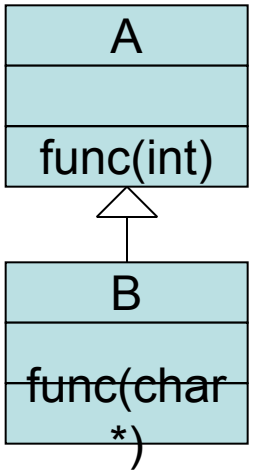
Пример:

```
#include <iostream>
int main()
{   std::cout << "Hello ";
    using namespace std;
    cout << "World." << endl;
}
```

Однако можно по-прежнему использовать определение ресурсов стандартных библиотек в глобальном пространстве. Для этого необходимо подключать <stdio.h>, <conio.h>, <math.h> и т.д. (кроме <iostream.h>).

Список доступных стандартных библиотек в старой и новой формах можно посмотреть в среде.

Использование пространств имен для перегрузки методов класса (Ex5_111)




```
#include "stdafx.h"
#include <conio.h>
class A
  {    public:    void func(int ch);
  };
class B : public A
  {    public:
          void func(char *str);
          using A::func;  // перегрузить B::func
  };
void A::func(int ch)     // метод базового класса
{ std::cout<<"Symbol\n"; }
void B::func(char *str) // метод производного класса
{ std::cout<<"String\n"; }
int main()
{   B b;
    b.func(25);      // вызов  A::func()
    b.func("ccc");   // вызов  B::func()
   getch();
   return 0;
}
```