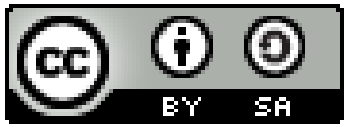
Пространства имён

(namespaces)

Это произведение доступно по лицензии
Creative Commons “Attribution-ShareAlike” (“Атрибуция — На тех же условиях”) 3.0 Непортированная.
http://creativecommons.org/licenses/by-sa/3.0/deed.ru



May 25, 2016

Проблема

В языке C только одна глобальная область видимости

Библиотека A: заголовочный файл liba.h

```
class Object { /* ... */ }; // 3-d объект
int show(void);  // отображение 3-d сцены
int version = 3;
```

Библиотека B: заголовочный файл libb.h

```
class Object { /* ... */ }; // элемент окна
int show(void);  // отображение окна
const char* version = "1.20.07";
```

Программа пользователя:

```
#include "liba.h"
#include "libb.h" // здесь уже есть конфликт имён
int main() {
  Object a; // неоднозначность: какой Object?
  show();   // неоднозначность: какая show()?
  return 0;
}
```

Решение проблемы методами C

Использование префиксов

Библиотека A: заголовочный файл liba.h

```
class gl3d_Object { /* ... */ }; // 3-d объект
int gl3d_show(void);  // отображение 3-d сцены
int gl3d_version = 3;
```

Библиотека B: заголовочный файл libb.h

```
class win_Object { /* ... */ }; // элемент окна
int win_show(void);  // отображение окна
const char* win_version = "1.20.07";
```

Программа пользователя

```
#include "liba.h"
#include "libb.h"
int main() {
  gl3d_Object o1; win_Object w;
  gl3d_show();    win_show();
  return 0;
}
```

Достоинства и недостатки подхода в стиле C

Достоинства:

- работает (если префиксы уникальные);
- не требует изменений в синтаксисе языка.

Недостатки:

- код программы загромождается префиксами;
- возникает проблема выборы короткого, но уникального префикса;
- у пользователя нет возможности управлять именами.

Библиотека A: заголовочный файл liba.h

```
namespace gl3d {
  class Object { /* ... */ }; // 3-d объект
  int show(void);  // отображение 3-d сцены
  int version = 3;
};
```

Библиотека B: заголовочный файл libb.h

```
namespace win {
  class Object { /* ... */ }; // элемент окна
  int show(void);  // отображение окна
  int status();
  const char* version = "1.20.07";
};
```

Библиотека A: файл liba.cpp

```
#include "liba.h"
gl3d::Object::Object()
{
   //  конструктор класса Object из
   //  пространства имён gl3d
}

int gl3d::show()
{
   //  функция show
   //  пространства имён gl3d
}
```

Программа пользователя: использование оператора ::

```
#include "liba.h"
#include "libb.h"
int show() { /* просто функция show */ }

int main()
{
  gl3d::Object o1;
  win::Object w;
  gl3d::show();
  win::show();
  ::show();  /* или просто */ show();
  cout << gl3d::version << ' ' << win::version;
  return 0;
}
```

Объявление using

Импорт одного имени в текущую область видимости

Программа пользователя: объявление using

```
#include "liba.h"
#include "libb.h"
int show() { /* просто функция show */ }

using gl3d::Object; //  видно до конца файла
int main()
{
  using win::show;   //  видно до конца блока
  Object o1;         //  из gl3d
  win::Object w;     //  указано явно
  gl3d::show();      //  указано явно
  show();            //  ОШИБКА - неоднозначность
  win::show();       //  из win
  ::show();          //  глобальная
  cout << gl3d::version << ' ' << win::version;
  return 0;
}
```

Директива using

Импорт всех имен в текущую область видимости

Программа пользователя: директива using

```
#include "liba.h"
#include "libb.h"

using namespace gl3d; //  видно до конца файла
                      //  - все имена из gl3d
int main()
{
  Object o1;          //  из gl3d
  win::Object w;      //  указано явно
  show();             //  из gl3d
  win::show();        //  из win
  cout << version << ' ' << win::version;
  return 0;
}
```

Конфликты имен
и их разрешение

Программа пользователя: несколько директив using

```
#include "liba.h"
#include "libb.h"

using namespace gl3d;
using namespace win;

int main()
{
  Object o1;          //  Ошибка - неоднозначность
  win::Object w;      //  указано явно
  show();             //  Ошибка - неоднозначность
  gl3d::show();       //  из gl3d
  status();           //  из win, так как нет в gl3d
  cout << gl3d::version << ' ' << win::version;
  return 0;
}
```

Неименованные (безымянные) пространства имён

Сокрытие реализации на уровне файла

Файл mylib.h

```
namespace {   //  безымянное пространство имён
   class A { };
   void funxx();
   int zzz;
};
//  <– здесь неявно срабатывает using namespace

A::A()
{}

void funxx()
{
   zzz = 0;
}
```

За пределами этого файла имена “A”, “funxx”, “zzz” не видны.

Пространства имён открыты

В отличие от классов, в пространствах имён нет модификаторов **public**, **private**, **protected**. Все имена из именованных пространств имён пространств имён доступны снаружи.

Пространства имён пополняемые

В любой момент пространство имён можно пополнить новыми определениями.

```
namespace A {
  int x;
};
namespace A {
  int y;
};
```

Псевдонимы пространств имён

Использование коротких имён вместо длинных

Для избежания конфликта имён среди самих пространств имён желательно использовать длинные имена. Пользоваться длинными именами неудобно. Решение: **Псевдонимы пространств имён** (синонимы, алиасы).

```
namespace my_lovely_graph_lib_v20 {
  class Window {}; /* ....... */
};

namespace MGL = my_lovely_graph_lib_v20;
//  теперь вместо my_lovely_graph_lib_v20
//  можно использовать MGL

MGL::Window w;
```

Нельзя пополнять пространства имён, используя псевдонимы.

Преимущества и недостатки пространств имён

Недостатки

- потребовался новый синтаксис языка;
- язык стал сложнее в изучении.

Преимущества

- имена в программе можно разделить на области применения;
- если не конфликта имён, при помощи “using” можно использовать короткую форму записи;
- при необходимости конфликт имён разрешается явно;
- можно использовать для сокрытия реализации, не находящейся в классе;
- пользователь контролирует, каким образом используются имена в программе.

Пространства имён и библиотеки

Во избежание конфликта имён современные библиотеки C++ помещаются в собственные пространства имён.

Стандартная библиотека C++ находится в пространстве имён “**std**”.

Для совместимости в языком C, стандартная библиотека C++ состоит из двух частей:

- старая часть, которая практически совпадает со стандартной библиотекой языка C. Доступны такие функции как *printf*, *open*, *sin*, . . .
- современной, объектно-ориентированной и шаблонной части.

Современная часть библиотеки C++ может использоваться только через пространство имён “std”.

```
#include <iostream>   //  включение заголовка
#include <string>
void  f()
{
   //  требуется явное указание std::
   std::string  s = "ABCDE";
   std::cout << s << std::endl;
}

//  импортируем все имена из пространства std
//  в текущую область видимости.
using namespace std;

void  g()
{
   string  s = "ABCDE";  //  теперь можно не указывать std::
   cout << s << endl;
}
```

Унаследованная часть библиотеки C++ может использоваться так, как использовалась в языке C. При этом включаются привычные заголовочные файлы, и импорт имён из std происходит автоматически.

```
#include <stdio.h>
#include <string.h>
//  имена автоматически импортированы из std

void f()
{
  const char *s = "ABCDE";
  int i = strlen( s );
  printf( "s= %s, i= %d\n", s, i );
}
```

Стандартная библиотека C++

Использование унаследованной части (2)

Унаследованная часть библиотеки C++ может использоваться так же, как и современная. При этом от имени заголовочного файла отбрасывается суффикс “.h” и добавляется префикс “c”.

```
#include <cstdio>
#include <cstring>
void f()
{
  const char *s = "ABCDE";
  int i = std::strlen( s );
  std::printf( "s= %s, i= %d\n", s, i );
}
using namespace std;
void g()
{
  const char *s = "ABCDE";
  int i = strlen( s );
  printf( "s= %s, i= %d\n", s, i );
}
```