Пространства имён

Александр Смаль

CS центр
19 марта 2015
Санкт-Петербург

Пространства имён

Пространства имён (namespaces) — это способ разграничения областей идентификаторов в C++.

Имена в C++:

1. имена переменных и констант,
2. имена функций,
3. имена структур и классов,
4. имена шаблонов,
5. синонимы типов (typedef-ы),
6. enum-ы и union-ы,
7. имена пространств имён.

Примеры

В C для избежания конфликта имён используются префиксы функций. К примеру, все имена в библиотеке Expat начинаются с `XML_`.

```
struct XML_Parser;
int XML_GetCurrentLineNumber(XML_Parser * parser);
```

В C++ это можно было бы записать так:

```
namespace XML {
    struct Parser;
    int GetCurrentLineNumber(Parser * parser);
}
```

Снаружи эта функция будет доступна как `XML::GetCurrentLineNumber`.

Описание пространст имён

1. Пространства имён могут быть вложенными:

```
namespace ru { namespace spb { namespace megacode {
    struct Array {...};
}}}
ru::spb::megacode::Array globalData;
```

2. Определение пространств имён можно разделять:

```
namespace en {
    int gcd(int a, int b) {...}
}
namespace gb {
    int hcf(int a, int b) { return en::gcd(a, b); }
}
namespace en {
    struct List {...};
}
```

3. Классы и структуры определяют одноимённый namespace.

Доступ к именам

1. Внутри того же `namespace` все имена доступны напрямую.
2. `NS::` позволяет обратиться внутрь пространства имён `NS`.

```
namespace NS {
    int foo() { return 0; }
}
int i = NS::foo();
```

3. Оператор `::` позволяет обратиться к *глобальному пространству имён*.

```
struct List {...};
namespace en {
    struct List {...};

    ::List globalList;
}
```

Поиск имён

Поиск имён — это процесс разрешения имени.

1. Если такое имя есть в текущем `namespace`, остановиться и выдать *все* одноимённые сущности в текущем `namespace`.
2. Если текущий `namespace` — глобальный, выдать ошибку.
3. Текущий `namespace` ← родительский `namespace`.
4. Перейти на шаг 1.

```
int foo(int i) { return 1; }
namespace ru {
    int foo(float f) { return 2; }
    int foo(double a, double b) { return 3; }
    namespace spb {
        int global = foo(5);
}}
```

Важно: поиск останавливается как только нашёл что-то. Перегрузка выполняется только для найденных имён.

Ключевое слово `using`

Существуют два различных использования слова `using`.
Добавление конкретного имени в текущий namespace:

```
void foo(int i) { return 1; }
namespace ru {
    using ::foo;
    int foo(float f) { return 2; }
    int foo(double a, double b) { return 3; }
    namespace spb {
        int global = foo(5);
}}
```

Ключевое слово `using` (продолжение)

Добавление всех имён одного `namespace` в текущий `namespace`:

```
namespace ru {
    namespace spb {
        int foo(int i) { return 1; }
    }
    namespace msk {
        using namespace spb;
        void foo(float f) { return 2; }
        int global = foo(5);
}}
```

Поиск Кёнига

```
namespace cg {
    struct Point2 {...};

    Point2 operator+(Point2 a, Point2 const& b);
}

int main() {
    cg::Point2 a(1,2);
    cg::Point2 b(3,4);
    b = a + b; // equivalent to: b = operator+(a, b)
    b = cg::operator+(a, b); // OK
    return 0;
}
```

Поиск Кёнига

```
namespace cg {
    struct Point2 {...};

    Point2 operator+(Point2 a, Point2 const& b);
}

int main() {
    cg::Point2 a(1,2);
    cg::Point2 b(3,4);
    b = a + b; // equivalent to: b = operator+(a, b)
    b = cg::operator+(a, b); // OK
    return 0;
}
```

Argument-dependent name lookup (ADL, Поиск Кёнига)

При поиске имени функции на первой фазе рассматриваются имена из текущего пространства имён *и пространств имён, к которым принадлежат аргументы функции*.

Безымянный namespace

Безымянный namespace — это пространство имён, имя которого уникально (генерируется компилятором).

```
namespace { // unnamed
    struct Test {
        std::string name;
    };
}
```

Это эквивалентно:

```
namespace $GeneratedName$ {
    struct Test {
        std::string name;
    };
}
using namespace $GeneratedName$;
```

Безымянные пространства имён — замена для `static`.

Заключение

1. Используйте пространства имён для исключения конфликта имён.
2. Помните, что поиск имён прекращается после первого совпадения. Используйте using и полные имена.
3. Не используйте `using namespace` в заголовочных файлах.
4. Всегда определяйте операторы в том же пространстве имён, что и типы, для которых они определены.
5. Используйте безымянные пространства имён для маленьких локальных утилитарных классов и как замену слова `static`.
6. Для длинных имён `namespace`-ов используйте синонимы:

```
namespace csccpp15 = ru::spb::csc::cpp15;
```