

# Программирование на языке C++

Шаблов Анатолий

[anatoliishablov@gmail.com](mailto:anatoliishablov@gmail.com)

ИТМО, весенний семестр 2024

# Пространства имён

# Пространства имён

Позволяют использовать одинаковые имена для разных сущностей в разных частях программы.

- Пространства имён и определения.
- Вложенные пространства имён.
- Область видимости пространства имён.
- Анонимные пространства имён.
- Полностью квалифицированные имена.
- Псевдонимы пространств имён.
- `using` директива.
- `inline`.

# Добавление в пространство std

В общем случае добавление в пространство std запрещено и является UB. Исключение: специализация шаблонных классов из пространства std, которая зависит хотя бы от одного пользовательского типа.

Ограничения добавления специализаций:

- Полная специализация метода класса из std – UB
- Полная или частичная специализация класса, являющегося вложенным в класс из std – UB

# Поиск имен

# Виды поиска имён

- Квалифицированный.
- Неквалифицированный.
- ADL.

# Квалифицированный поиск имён

Квалифицированное имя – имя справа от оператора :: .

Может относиться к:

- Члену класса.
- Сущности из пространства имён.
- Элементу enum.

Слева от оператора :: – либо ничего, либо уже найденное имя (класса, пространства имён или enum).

# Неквалифицированный поиск

Если имя не стоит справа от :: применяется неквалифицированный поиск.

При таком поиске будут анализироваться только подходящие сущности (например, для имени слева от :: – только пространства имён, классы и enum).

Такой поиск обрабатывает using директивы, как если бы содержимое соответствующего пространства имён находилось в этом пространстве имён. При этом поиск в using обрабатывается уже после основного поиска

Поиск будет идти вплоть до обнаружения хотя бы одной сущности с таким именем или до исчерпания вариантов. Результатом будут все подходящие сущности с искомым именем, объявленные на первом уровне, на котором обнаружилось имя.



# ADL: пример

```
std::cout << x << std::endl;
```

```
operator << (std::cout, x);
```

```
operator << (std::cout, std::endl);
```

```
std::cout << endl; // error: 'endl' identifier not found
```

```
endl(std::cout); // OK
```

# ADL: правила

Применяется для имени из выражения вызова функции.

Не производится, если неквалифицированный поиск дал:

- Член класса.
- Объявление функции на уровне блока (не считая using объявления).
- Объявление сущности, не являющееся функцией.

Иначе каждый аргумент функционального вызова добавляет пространства имён и/или классы во множество, в котором затем будет производиться поиск (помимо обычного неквалифицированного поиска).

В процессе поиска отбираются только объявления функций.

# ADL: правила для различных типов аргументов

В зависимости от типа аргумента к поиска добавляется:

- Базовый тип – ничего.
- Класс – сам класс, его предки, его область определения (окружающий класс или пространство имён).  
Указатель на функцию – анализ применяется для типа возвращаемого значения и типов всех параметров.
- Шаблон – в дополнение к обычным правилам, анализ применяется для каждого шаблонного параметра-типа.
- `enum` – область определения `enum`.
- $T^*$  – анализ применяется для  $T$ .
- Указатель на метод – анализ применяется для класса, типа возвращаемого значения и всех параметров.
- Указатель на поле – анализ применяется для типа поля и класса.