Семинар 1.11 «Семантика перемещения в шаблонах»

- технически, одно из самых ключевых внедрений в C++11 (для шаблов - специальные правила и особенности)

main → f → g (передаём объект через посредника)

↑ ↑

требуется сохранение фундаментальных св-в - модифицируемость, перемещаемость, const

---

Пример move.cpp - экспоненциальный рост

```
template < typename T>
void f (T& v)
{
  g(v); // проблема с move
}
```

Perfect forwarding (идеальная передача):

```
template < typename T>
void f (T&& v) // пробрасывающая ссылка (универсальная?)
{
  g(std::forward<T>(v)); // условное приведение к rvalue&&
}
```

X&& - конкретный тип (rvalue ссылка)
T&& - выводимый тип (пробрасывающая ссылка)

выводится компилятором

std::move выполняет безусловное приведение к rvalue&&

Шаблоны специальных функций-членов (конструкторы)

Пример constructors.cpp - заготовка для идеальной передачи

Проблема с копирующим конструктором - нет const!

Отключение шаблонов с помощью std::enable_if:

- как он работает?
- как использовать? (см. PDF-приложение) + кортежи?

Идиома SFINAE - игнорирование шаблона, если он не подходит

std::enable_if + std::is_convertible_v решают проблему конструктора

Передача по значению или по ссылке? (в шаблонах)

- по умолчанию - по значению
  - выполняется копирование (но есть оптимизации)

- можно использовать std::ref(...) и std::cref(...)
- безопаснее для возвращаемого значения
- есть извлечение типа для обёртки
- передача по ссылке
  - лучшая производительность - но надо измерять!
  - модификация исходных объектов
  - пробрасывающий шаблон

Повторить правила свёртывания ссылок

```
template< typename T>
void f(T& arg); // с const также просто

int x = 42;          -> T ≡ int       , arg - int&
const int cx = x;    -> T ≡ const int , arg - const int&
const int& rx = x;   -> T ≡ const int , arg - const int&

template <typename T>
void f(T&& arg);

int x = 42;          -> T ≡ int&      , arg ...
const int cx = x;    -> T ≡ const int&, arg ...
const int& rx = x;   -> T ≡ const int&, arg ...
    f(42);           -> T ≡ int,       arg - int&&

template< typename T>
void f(T arg);

auto v = rx; // int
decltype(auto) v = rx; // const int&
```

Все тривиально.

initializer_list! ←

Все это аналогично применимо и к выводу типа auto.

Реализация шаблонов св-в и преобразований типов

...v -> bool - проверяющие (is_same_v)
...t -> type - создающие (common_type_t)

Примеры (live)

- add / remove reference
- is_same, is_array
- is_derived (Г. Саттер / А. Александреску)
- if-then-else