### Приведение типов

Гусев Илья

Московский физико-технический институт

Москва, 2017

## Содержание

- 🚺 Приведение типов
  - static cast
  - const\_cast
  - dynamic cast
  - reinterpret cast
  - C-style cast

## static cast

#### Можно:

- Преобразование указателя на родительский класс к указателю на дочерний класс. Объект по указателю обязан быть правильного дочернего класса, иначе undefined behaviour.
- Преобразования между числовыми типами. int, long, char, unsigned int все их можно кастить друг в друга при помощи static\_cast.
- Можно закастить любое выражение в void (ради побочных эффектов).
- Можно привести константу nullptr к любому типу-указателю.

#### Нельзя:

- Преобразование между указателями на в принципе несовместимые типы.
   Например, указатель на double нельзя привести к указателю на int.
- Указатели на типы, а также сами типы с несовместимыми атрибутами const и/или volatile.
- Привести указатель на функцию-член к указателю на обычную функцию, или указатель на код к указателю на данные.



# static\_cast - примеры

```
// Пример числовых преобразований.
double a = 1.0;
int b = static_cast<int>(a);

// Пример каста вверх по иерархии.
class A {};
class B: public A{};
B b;
A* a = static_cast<A*>(&b);

// Пример попытки снять константность.
const int a = 1;
int b = static_cast<int>(a); // Ошибка при компиляции!
```

### const cast

- Добавляет или убирает модификатор const/volatile у переменной.
- Если вы изначально определили переменную как const, нельзя на ней использовать const\_cast, иначе UB!
- Оспользуется для вызовов правильных перегрузок, например.
- Добавить const к неконстантному объекту могут также static\_cast и dynamic cast.

#### const cast - примеры

```
int f(const int* a) {
    int* b = const_cast < int* > ( a );
    (*b) ++;
    return *b;
}

const int c = 1;
int i = 1;

f(c); // UB.
f(i); // Корректно.
```

# dynamic cast

- Нужен для полиморфизма, используется только с классами с виртуальными функциями или виртуальным наследованием.
- Если преобразование по иерархии действительно возможно (подходящий изначальный объект), оно произойдёт.
- Если преобразование невозможно, и преобразовывались указатели, dynamic\_cast вернёт 0 или nullptr.
- О Если преобразование невозможно, и перобразовывались ссылки, dynamic\_cast кинет исключение std::bad\_cast.

# dynamic\_cast - примеры

```
struct A {
     virtual ~A() {}
};
struct B : A {
     int b = 1;
};

A a;
B b;
A* dynamic_b_a = dynamic_cast <A*>(&b); // OK.
A* static_b_a = static_cast <A*>(&b); // OK.
B* dynamic_a_b = dynamic_cast <B*>(&a); // O.
B* static_a_b = static_cast <B*>(&a); // UB.
static_a_b->b; // Mycop.
```

## reinterpret\_cast

- Самое опасное преобразование, ипользуется для грязных хаков.
- Напрямую превращает один тип в другой.
- Ограничения: только pointer-to-pointer, reference-to-reference, pointer-to-integer, integer-to-pointer преобразования.
- Гарантирует, что если преобразовать сначала в одну сторону, а потом обратно, то получится то же значение. Исключение - если размер типа, в который преобразовываем, меньше исходного.

## reinterpret\_cast

```
int a = 1;
// Верхние 4 байта - мусор.
double b = *reinterpret_cast < double *>(&a); // Случайное значение.

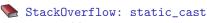
if (sizeof(float) == sizeof(int)) {
   int a = 1;
   // Конкретное значение зависит от платформы.
   // Размер типов, порядок байтов.
   float b = *reinterpret_cast < float *>(&a); // 1.4013e-45
}
```

### C-style cast

#### Порядок вызова:

- const cast.
- ② Если const\_cast не может дать нужный результат, то static\_cast (с игнорированием спецификатора приватного наследования).
- Если и так не выходит, то компилятор пробует в хвост к static\_cast добавить const cast.
- О Если и это не получается, то reinterpret \_cast.
- ⑤ Если не выйдет, то к нему дописывается const\_cast.

#### Полезные ссылки І



https://ru.stackoverflow.com/questions/387985/Для-чегонужен-static-cast-как-он-работает-и-где-его-применяют



StackOverflow: When should static\_cast, dynamic\_cast, const\_cast and reinterpret\_cast be used? https://stackoverflow.com/questions/332030/when-should-static-cast-dynamic-cast-const-cast-and-reinterpret-cast-be-used