

[КАК СТАТЬ АВТОРОМ](#)[Карьерные консультации](#)[Опрос: какие темы вы хотите вид...](#)

rpz 13 декабря 2013 в 14:21

## Секреты тернарного оператора

C++ \*, C \*

Каждый уважающий себя программист C\C++ знает что такое тернарный оператор и большинство использовало его хотя бы раз в своих программах. Но знаете ли вы все секреты тернарного оператора? Какие потенциальные опасности сопряжены с его использованием и какие, казалось бы не связанные с его прямым предназначением, возможности в нем таятся? Эта статья дает вам возможность проверить свои знания и, возможно, узнать что-то новое.

Начнем с небольшого теста.

### Тест

Скомпилируется ли следующий код? Объясните почему.

1.

```
int i;  
int j;  
(false ? i: j) = 45;
```

2.

```
int i;  
int j;  
(true ? i: j) = 45;
```

3.

```
short i;  
int j;  
(true ? i: j) = 45;
```

4.

```
return true ? 0 : 1;
```

5.

```
true ? return 0 : return 1;
```

Какой будет вывод у следующего кусочка? Почему?

6.

```
std::cout << (false ? 9 : '9') << " " << (true ? 9 : '9');
```

Какие значения будут у переменных `a`, `b` и `c` в результате выполнения следующего кода? Почему?

7.

```
int a = 1;
int b = 1;
int c = 1;
a = true ? ++b : ++c;
```

8. Назовите ситуацию, где нельзя использовать `if{...} else{...}`, но можно тернарный оператор.
9. Какие потенциальные опасности скрываются в использовании тернарного оператора? В чем их причина?
10. Какие неожиданные использования тернарного оператора приходят вам в голову?

## Объяснение

Итак, начнем. Тернарный оператор выделяется из ряда других операторов в C++. Его называют "*conditional expression*". Ну а так как это *expression*, выражение, то как у каждого выражения, у него должен быть тип и *value category*. Собственно, ответив на вопросы какой тип и *value category* у тернарных операторов в каждом из первых семи вопросов теста, мы легко решим поставленные задачи.

Здесь начинается самое интересное. Оказывается типом тернарного оператора будет *наиболее общий* тип его двух последних операндов. Что значит *наиболее общий*? Это легче всего пояснить на примерах. У `int` и `short` общим типом будет `int`.

У `A` и `B` в следующем фрагменте общим типом будет также `int`.

```
struct A{ operator int(){ return 1; } };  
struct B{ operator int(){ return 3; } };
```

Т.е. *наиболее общий* тип это такой тип, к которому могут быть приведены оба операнда. Вполне могут быть ситуации, когда общего типа нет. Например у

```
struct C{};  
struct D{};
```

общего типа нет, и следующий фрагмент вообще не скомпилируется

```
(true ? C() : D());
```

Так. С типом тернарного оператора мы немного разобрались. Осталось решить вопрос с *value category*. Тут действует следующее правило: если в тернарном операторе происходит преобразование типов к *наиболее общему*, то тернарный оператор – *rvalue*. Если же нет, то *lvalue*. Теперь когда мы знаем то, что мы знаем, мы легко ответим на первые 7 вопросов.

## Ответы

1. и 2. – Да. Преобразования типов не происходит, а *lvalue* вполне можно присваивать значение.
3. – Нет. Здесь происходит преобразование типов. Значит *value category* у выражения слева от знака "=" – *rvalue*. А *rvalue*, как известно, нельзя присваивать.
4. – Да. Все мы так делали не раз.
5. – Нет. Здесь все дело в том, что в C++ *statement* не может разбивать *expression*.
6. Программа выведет «57 9». В данном фрагменте из-за того, что 2ой и 3ий операнд имеют разные типы, происходит преобразование к наиболее общему типу. В данном случае *int*. А '9', как известно, имеет ASCII код 57.
7. В этом вопросе кроется еще одна особенность тернарного оператора. А именно, вычисляется только тот операнд из второго и третьего, до которого доходит поток выполнения. Впрочем такое же поведение можно наблюдать у `if{...}else{...}`. Соответственно, значения переменных `a`, `b` и `c` будут 2, 2, 1.

## Где нельзя использовать `if{...} else{...}`, но можно тернарный оператор?

Например, в списке инициализации конструктора. Вы не можете написать так:

```
struct S
{
    S() : if(true) i_(1) else i_(0){}
    int i_;
};
```

Но вполне можно вот так:

```
struct S
{
    S() : i_(some_condition ? 0 : 1){}
    int i_;
};
```

При инициализации ссылки в зависимости от условия. Как известно, нельзя объявлять не инициализированную ссылку, поэтому следующий фрагмент не скомпилируется:

```
int a = 3;
int b = 4;
int& i;
if(some_condition)
    i = a;
else
    i = b;
```

А вот следующий скомпилируется успешно:

```
int& i = (some_condition ? a : b);
```

В C++11 тернарный оператор применяется гораздо чаще. Связано это с тем, что в *constexpr* функциях не должно быть ничего кроме *return 'expression'*. А *'expression'* вполне может представлять из себя тернарный оператор.

В качестве примера приведу классический алгоритм определения простоты числа

```
constexpr bool check_if_prime_impl(unsigned int num, unsigned int d)
{
    return (d * d > num) ? true :
```

```
        (num % d == 0) ? false :  
  
        check_if_prime_impl(num, d + 1);  
    }
```

Все потоки   Разработка   Администрирование   Дизайн   Менеджмент   Маркетинг   Научпоп



```
return (num <= 1) ? false :  
    check_if_prime_impl(num, 2);  
}
```

В этом же примере, кстати, видно использование каскадных тернарных операторов, которые могут быть неограниченной вложенности и заменять собой множественные `if{...} else{...}`.

## Опасности тернарного оператора

Допустим у нас есть класс **String**

```
class String  
{  
    public:  
    operator const char*();  
};
```

И использовать мы его можем, например, так:

```
const char* s = some_condition ? "abcd" : String("dcba");
```

Как нам уже известно, второй и третий операнд тернарного оператора приводятся к наиболее общему типу. В данном случае это **const char\***. Но объект **String(«dcba»)** уничтожится в конце выражения и **s** будет указывать на невалидную память. В лучшем случае программа упадет при попытке в дальнейшем использовать **s**. В худшем будет выдавать неверные результаты, вызывая недовольство у заказчика и головную боль у программиста.

## «Необычное» использование тернарного оператора

Тернарный оператор можно использовать для определения общего типа двух и более типов. А это, в свою очередь, можно использовать, например, для определения приводится ли один тип к другому.

```
template <typename T, typename U>
```

```
template<typename T, typename U>  
struct common_type  
{  
    typedef decltype(true ? std::declval<T>() : std::declval<U>()) type;  
};  
  
template<typename T, typename U>  
struct is_same{ enum { value = false; } };  
  
template<typename T>  
struct is_same<T, T>{ enum { value = true; } };  
int main()  
{  
    std::cout << is_same<int, common_type<A, B>::type>::value <<std::endl;  
}
```

На самом деле, если знать свойства тернарного оператора, такое использование практически напрашивается само собой. Необычным здесь, пожалуй является лишь то, что он используется не по прямому назначению, т.е. не для выбора одного значения из двух в зависимости от условия.

Теги: [C++](#), [C](#)

Хабы: [C++](#), [C](#)

## Редакторский дайджест

Присылаем лучшие статьи раз в месяц

**76**

Карма

**0**

Рейтинг

**Куликов Александр** @grpz

Пользователь

Реклама

РЕКЛАМА

**Точка**

для ИП и ООО до 31 марта  
**Счёт полгода — бесплатно**

[Открыть](#)

 Комментарии 39

## ПОХОЖИЕ ПУБЛИКАЦИИ

29 июля 2019 в 16:03

**C++20 укомплектован, C++23 начат. Итоги встречи в Кёльне** +64 33K 77 348 +348

13 июня 2019 в 18:39

**Лямбды: от C++11 до C++20. Часть 2** +41 21K 130 29 +29

20 марта 2019 в 17:26

**Лямбды: от C++11 до C++20. Часть 1** +24 38K 201 8 +8

## МИНУТОЧКУ ВНИМАНИЯ

Разместить



Промо

**Образование будущего: всё будет хорошо, но непривычно**Опрос **Хотите рассказать о себе в наших социальных сетях?**

## ЗАКАЗЫ

Автозаполнение формы на сайте из MS Excel (7 текстовых полей)

10000 руб./за проект • 5 откликов • 30 просмотров

Сверстать пару страниц мобильного приложения (Xamarin)

2000 руб./за проект • 1 отклик • 19 просмотров

Разработка телеграмм бота на Python с учётом технол

50000 руб./за проект • 14 откликов • 91 просмотр

Улучшение WebBrowser виджет в UE4

40000 руб./за проект • 1 отклик • 15 просмотров

Написание бэка для сайта на React JS

40000 руб./за проект • 5 откликов • 45 просмотров

Больше заказов на Хабр Фрилансе

## ЛУЧШИЕ ПУБЛИКАЦИИ ЗА СУТКИ

вчера в 13:30

**Как в декабре я устроился в \$βΣP\*, а в марте меня уволили одним днем**

◆ +82

👁 59K

🔖 51

💬 519 +519

вчера в 16:00

**Профессиональный обман: как мы рассылаем фишинговые письма нашим клиентам**

◆ +47

👁 7.3K

🔖 24

💬 13 +13

вчера в 16:00

**Сам себе РКН или родительский контроль с MikroTik (ч.3)**

◆ +25

👁 3.3K

🔖 71

💬 10 +10

сегодня в 01:44

**Золотые лампочки Navigator Supervision**

◆ +22

👁 5.3K

🔖 22

💬 15 +15

вчера в 14:04

**Распределённые транзакции Kafka + PostgreSQL средствами Spring**

◆ +21

👁 2.1K

🔖 50

💬 9 +9

**Четыре слагаемых успешной системы образования**

Интересно



ЧИТАЮТ СЕЙЧАС

Как в декабре я устроился в \$βΣP\*, а в марте меня уволили одним днем

59K 519 +519

Как мы сами себя ловим в карьерной ловушке

13K 8 +8

Осталось 3 дня, чтобы что-то сделать со школой, если вы уехали из России

8.5K 12 +12

Исследователи обнаружили «фабрику» вредоносных пакетов prn

7.1K 16 +16

Золотые лампочки Navigator Supervision

5.3K 15 +15

Почему программисты пишут статьи?

Мегапост

РАБОТА

Программист C  
38 вакансий

Программист C++  
114 вакансий

QT разработчик  
14 вакансий

Все вакансии

Реклама

РЕКЛАМА





Денежные переводы

из России за рубеж

Подробности на [koronapay.com](https://koronapay.com)

УСЛУГА ПРЕДОСТАВЛЯЕТСЯ РНКО «ПЛАТЕЖНЫЙ ЦЕНТР» (ООО), Г. НОВОСИБИРСК, УЛ. КИРОВА, 96, ОГРН 1025400002968, ЛИЦЕНЗИЯ ЦБ РФ № 3166-К ОТ 14.04.2014 Г. ПЕРЕЧЕНЬ НАПРАВЛЕНИЙ, ОГРАНИЧЕНИЙ, ДОГОВОРЫ О КОМ- ПЛЕКСНОМ ОБСЛУЖИВАНИИ КЛИЕНТОВ, А ТАКЖЕ УСЛОВИЯ ОКАЗАНИЯ УСЛУГ- ГИ СМОТРИТЕ НА [RNCB.RU](https://rncb.ru) В РАЗДЕЛЕ «ДЕНЕЖНЫЕ ПЕРЕВОДЫ». РНКО «ПЛАТЕЖНЫЙ ЦЕНТР» (ООО) ВПРАВЕ ОТКАЗАТЬ В ОКАЗАНИИ УСЛУГИ.

Ваш аккаунт	Разделы	Информация	Услуги
Войти	Публикации	Устройство сайта	Реклама
Регистрация	Новости	Для авторов	Тарифы
	Хабы	Для компаний	Контент
	Компании	Документы	Семинары
	Авторы	Соглашение	Мегапроекты

<https://habr.com/ru/post/205848/>

10/11

Песочница

Конфиденциальность



Настройка языка

0 сайте

Техническая поддержка

Вернуться на старую версию

© 2006–2022 «Habr»