The New C Standard (Excerpted material)

An Economic and Cultural Commentary

Derek M. Jones
derek@knosof.co.uk

6.3.1.2 Boolean type

_Bool converted to

When any scalar value is converted to **_Bool**, the result is 0 if the value compares equal to 0; 680

Commentary

Converting a scalar value to type **_Bool** is effectively the same as a comparison against 0; that is, `(_Bool)x` is effectively the same as `(x != 0)` except in the latter case the type of the result is **int**.

Conversion to **_Bool** is different from other conversions, appearing in a strictly conforming program, in that it is not commutative— `(T1)(_Bool)x` need not equal `(_Bool)(T1)x`. For instance:

```
(int)(_Bool)0.5 ⇒ 1
(_Bool)(int)0.5 ⇒ 0
```

Reordering the conversions in a conforming program could also return different results:

```
(signed)(unsigned)-1 ⇒ implementation-defined
(unsigned)(signed)-1 ⇒ UINT_MAX
```

C90

Support for the type **_Bool** is new in C99.

C++

4.12p1 *An rvalue of arithmetic, enumeration, pointer, or pointer to member type can be converted to an rvalue of type* ***bool****. A zero value, null pointer value, or null member pointer value is converted to* ***false****;*

The value of **false** is not defined by the C++ Standard (unlike **true**, it is unlikely to be represented using any value other than zero). But in contexts where the integer conversions are applied:

4.7p4 *. . . the value* ***false*** *is converted to zero . . .*

Other Languages

Many languages that include a boolean type specify that it can hold the values true and false, without specifying any representation for those values. Java only allows boolean types to be converted to boolean types. It does not support the conversion of any other type to boolean.

Coding Guidelines

The issue of treating boolean values as having a well-defined role independent of any numeric value is discussed elsewhere; for instance, treating conversions of values to the type **_Bool** as representing a change of role, not as representing the values 0 and 1. The issue of whether casting a value to the type **_Bool**, rather than comparing it against zero, represents an idiom that will be recognizable to C developers is discussed elsewhere.

boolean role

boolean role

otherwise, the result is 1. 681

Commentary

if statement operand compare against 0

In some contexts C treats any nonzero value as representing true — for instance, controlling expressions (which are also defined in terms of a comparison against zero). A conversion to **_Bool** reduces all nonzero values to the value 1.

C++

4.12p1

. . . ; any other value is converted to **`true`**.

The value of **`true`** is not defined by the C++ Standard (implementations may choose to represent it internally using any nonzero value). But in contexts where the integer conversions are applied:

. . . the value **`true`** *is converted to one.* 4.7p4

References