Integer division rounding with negatives in C++

Asked 16 years ago Modified 7 years ago Viewed 27k times

44

Suppose `a` and `b` are both of type `int`, and `b` is nonzero. Consider the result of performing `a/b` in the following cases:

1. `a` and `b` are both nonnegative.
2. `a` and `b` are both negative.
3. Exactly one of them is negative.

In Case 1 the result is rounded down to the nearest integer. But what does the standard say about Cases 2 and 3? An old draft I found floating on the Internet indicates that it is implementation dependent (yes, even case 2) but the committee is leaning toward making it always 'round toward zero.' Does anyone know what the (latest) standard says? Please answer only based on the standard, not what makes sense, or what particular compilers do.

c++ **rounding**

Share

Improve this question

edited May 21, 2009 at 14:10

17 of 26

27.4k ● 13 ● 68 ● 85

Follow

asked Nov 26, 2008 at 6:06

Ari

2 Incredible research opportunity given the 1200 page nature of the standard. I'm going to give it a quick grep and give up :) – Stefan Mai Nov 26, 2008 at 6:14

Seeing as this question still pops up first in search results, is there any updated answer that could be linked in for c++ 14 or 17, etc.? – mattgately Feb 29 at 17:38

4 Answers

Sorted by: Highest score (default)

32

As an update to the other answers:

The last draft of C++11, n3242 which is for most practical purposes identical to the actual C++11 standard, says this in 5.6 point 4 (page 118):

> For integral operands the / operator yields the algebraic quotient with any fractional part discarded; (see note 80)

Note 80 states (note that notes are non-normative):

> 80) This is often called truncation towards zero.

Point 4 goes on to state:

> if the quotient a/b is representable in the type of the result, (a/b)*b + a%b is equal to a.

which can be shown to require the sign of `a%b` to be the same as the sign of `a` (when not zero).

Share Improve this answer Follow

edited Nov 16, 2011 at 1:45

answered Nov 15, 2011 at 14:08

Sjoerd

6,865 ● 32 ● 45

26

According to the May 2008 revision,

You're right:

> The binary / operator yields the quotient, and the binary % operator yields the remainder from the division of the first expression by the second. If the second operand of / or % is zero the behavior is undefined; otherwise (a/b)*b + a%b is equal to a. If both operands are nonnegative then the remainder is nonnegative; if not, the sign of the remainder is implementation-defined75).

Note 75 says:

> According to work underway toward the revision of ISO C, the preferred algorithm for integer division follows the rules defined in the ISO Fortran standard, ISO/IEC 1539:1991, in which the quotient is always rounded toward zero.

Chances are that C++ will lag C in this respect. As it stands, it's undefined but they have an eye towards changing it.

I work in the same department as Stroustrup and with a member of the committee. Things take AGES to get accomplished, and its endlessly political. If it seems silly, it probably is.

Share Improve this answer

Follow

answered Nov 26, 2008 at 6:24

Stefan Mai

23.9k ● 7 ● 57 ● 62

12 The quoted statement is old. It dates back to the C++98 standard and refers to the C99 revision. C99 specifies rounding toward zero and C++11 follows suit. – Jed May 29, 2012 at 14:30

9

Just a comment. The current working draft for the C++ standard indeed corrects the "implementation-defined" issue and asks for truncation towards zero. Here is the committee's webpage, and here is the draft. The issue is at page 112.

Share Improve this answer

Follow

answered Nov 26, 2008 at 6:35

Federico A. Ramponi

47k • 30 • 111 • 133

-7

Sometimes we need to take a step back, and look just at the mathematics of it:

Given int x, int y

if int i1 = x/y and int i2 = x%y

then y * i1 + i2 must be x

So this is not so much about the standard, but there is only one way this can possibly be. If any standards allows it to be any other way, then the standard is wrong, and that means the language is broken.

Share Improve this answer

Follow

answered Nov 23, 2017 at 20:28

user3283405

1 • 1

1 What you say is true, but it does not answer the question. Python and C++ define integer division and modulo consistent with what you say, but one uses floor division and the other round-toward-zero division, and so they do not exhibit the same behavior. – Kevin Feb 20, 2020 at 14:51

There is more than one way that this can be true. This question is also about the value for `x % y`. One can demand that, for positive y, `0 <= x % y < y`, which is what you probably have in mind and which I would prefer. However, as a correct answer shows, C++11 actually has, for

positive y, `0 <= | x % y | < y` and sign of x equals sign of `x % y` (if non-zero), – Carsten S Mar 13, 2020 at 15:28