Floating point comparison `a != 0.7` [duplicate]

Possible Duplicate:
problems in floating point comparison

```
#include <stdio.h>
#include <conio.h>

main()
{
    float a = 0.7;
    if(a < 0.7)
        printf("C");
    else
        printf("C++");
}
```

In the above code, the output is `C`. I tried this code in Code::Blocks and Pelles C but got the same answer. I would like know the reason for this in detail!

c comparison floating-point

edited May 10 '12 at 12:37
mskfisher
1,951 ●2 ●20 ●37

asked Jul 30 '11 at 13:12
Ashutosh Dave
221 ●8 ●24

marked as duplicate by Kev Aug 11 '11 at 17:29

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

6 Have a look at the floating point guide and what every computer scientist should know about floating-point arithmetic – Bart Jul 30 '11 at 13:19

3 Answers

In binary, 0.7 is:

```
b0.1011001100110011001100110011001100110011001100110011001100110...
```

However, `0.7` is a double-precision literal, whose value is 0.7 rounded to the closest representable double-precision value, which is:

```
b0.10110011001100110011001100110011001100110011001100110
```

In decimal, that's exactly:

```
 0.6999999999999999555910790149937383830547332763671875
```

When you write `float a = 0.7`, that double value is rounded again to single-precision, and `a` gets the binary value:

```
b0.101100110011001100110011
```

which is exactly

```
 0.699999988079071044921875
```

in decimal.

When you do the comparison `(a < 0.7)`, you are comparing this single-precision value (converted to double, which does not round, because all single-precision values are representable in double precision) to the original double-precision value. Because

```
 0.699999988079071044921875 <
0.6999999999999999555910790149937383830547332763671875
```

the comparison correctly returns true, and your program prints `"C"`.

Please note that none of this is any different in C++, appearances of the code in question to the contrary. There are certain (numerically unsafe) compiler optimizations that can change the behavior, but those are not unique to C or C++.

edited Jul 30 '11 at 18:32

answered Jul 30 '11 at 18:07
Stephen Canon
76.8k ●11 ●126 ●218

good demontstration, +1 – unkulunkulu Jul 30 '11 at 18:46

ok...thank you very much!! it helped a lot!! – Ashutosh Dave Jul 31 '11 at 11:25

It's because `0.7` has type `double`, so `a` gets converted to `double` and comparison is made in this type. As `0.7` is not representable exactly in binary floating-point, you get some rounding error and the comparison becomes true.

You can:

```
if( a < 0.7f ) {
....
```

But actually this effect is true for C++ too, so your conditional is not exactly rightful.

edited Jul 30 '11 at 13:30 answered Jul 30 '11 at 13:16
unkulunkulu
6,834 ●2 ●18 ●44

Ok..thanx a lot!! – Ashutosh Dave Jul 31 '11 at 11:24

Bart gave a very good reference in his comment, but I would also recommend this very simple rule:

Not all numbers can be represented exactly in the computer, so as soon as you use floating point numbers (such as float and double), you should expect that there can be a small, unpredictable error in each stored number. No, it isn't really random or unpredictable, but until you know more about it you can consider it unpredictable. Therefore, a copmparison such as your **a<0.7** might turn out to be true, and it might not. Don't write code that depends on it being either.

answered Jul 30 '11 at 14:32
Thomas Padron-McCarthy
19.5k ●4 ●33 ●62

I never used such comparisons but what if I have to compare some floating point values?? – Ashutosh Dave Jul 31 '11 at 11:27