Comparison of float and double variables [duplicate]

Possible Duplicates:
Difference between float and double
strange output in comparision of float with float literal

I am using visual C++ 6.0 and in a program I am comparing float and double variables For example for this program

```
#include<stdio.h>
int main()
{
    float a = 0.7f;
    double b = 0.7;
    printf("%d %d %d",a<b,a>b,a==b);
    return 0;
 }
```

I am getting 1 0 0 as output

and for

```
#include<stdio.h>
int main()
{
    float a = 1.7f;
    double b = 1.7;
    printf("%d %d %d",a<b,a>b,a==b);
    return 0;
 }
```

I am getting 0 1 0 as output.

Please tell me why I am getting these weird output and is there any way to predict these outputs on the same processor. Also how comparison is done of two variables in C ?

c comparison floating-point double

edited May 11 '12 at 18:05
mskfisher
1,961 ●2 ●21 ●37

asked Oct 21 '10 at 14:55
Chetan Sharma
21 ●1 ●1 ●2

marked as duplicate by msw, Péter Török, N 1.1, FrustratedWithFormsDesigner, Hans Passant Oct 21 '10 at 15:10

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.

4 You should never compare floating point values for equality - due to known precision and rounding issues, the results are unpredictable. And this is a duplicate of many posts - will look up one soon. – Péter Török Oct 21 '10 at 14:58

4 Dupe of hundreds of "Are floating point numbers broken?" questions... read those until you get **that floating point numbers are inherently inaccurate**. – delnan Oct 21 '10 at 14:58

2 @delnan: I wouldn't describe floating point numbers themselves as inherently inaccurate. I would describe the conversion from decimal values to binary floating point values as inherently lossy. – Jon Skeet Oct 21 '10 at 15:03

1 related stackoverflow.com/questions/2743718/when-is-aa-true/… :) – N 1.1 Oct 21 '10 at 15:07

1 @Jon Skeet: Yes, there are a lot of numbers that can be represented with floats. Sadly, there's an infinite number of rationals, and while juggling floats, one is almost guaranteed to run into those (i.e. inaccuracies). So one is better of treating floats as *always* inaccurate and consider 100% accurate results an exception :) – delnan Oct 21 '10 at 15:07

4 Answers

It has to do with the way the internal representation of floats and doubles are in the computer. Computers store numbers in binary which is base 2. Base 10 numbers when stored in binary may have repeating digits and the "exact" value stored in the computer is not the same.

When you compare floats, it's common to use an epsilon to denote a small change in values. For example:

```
float epsilon = 0.000000001;
float a = 0.7;
double b = 0.7;

if (abs(a - b) < epsilon)
  // they are close enough to be equal.
```

answered Oct 21 '10 at 15:01
Starkey
8,025 ●5 ●24 ●46

6 Wrong. You should use fabs instead. – user411313 Oct 21 '10 at 21:19

1.7d and 1.7f are very likely to be different values: one is the closest you can get to the absolute value 1.7 in a double representation, and one is the closest you can get to the absolute value 1.7 in a float representation.

To put it into simpler-to-understand terms, imagine that you had two types, `shortDecimal` and `longDecimal`. `shortDecimal` is a decimal value with 3 significant digits. `longDecimal` is a decimal value with 5 significant digits. Now imagine you had some way of representing pi in a program, and assigning the value to `shortDecimal` and `longDecimal` variables. The short value would be 3.14, and the long value would be 3.1416. The two values aren't the same, even though they're both the closest representable value to pi in their respective types.

edited Oct 21 '10 at 15:04

answered Oct 21 '10 at 14:58
Jon Skeet
916k ●502 ●6647 ●7551

He *is* passing integers. – Hans Passant Oct 21 '10 at 15:01

@Hans: The question changed after posting. It was *originally* printing `a` and `b` (nearly; actually `ab` which I assumed was a typo for `a,b`; it turns out it was just unformatted code, which made everything unclear). Will edit. – Jon Skeet Oct 21 '10 at 15:03 ✎

@Shynthriir: Look at the question pre-edit... – Jon Skeet Oct 21 '10 at 15:04

1 Apologies, I knew there had to be a reason. – Anthony Oct 21 '10 at 15:09

1.7 is decimal. In binary, it has non-finite representation.

Therefore, 1.7 and 1.7f differ.

Heuristic proof: when you shift bits to the left (ie multiply by 2) it will in the end be an integer if ever the binary representation is “finite”.

But in decimal, multiply 1.7 by 2, and again: you will only obtain non-integers (decimal part will cycle between `.4`, `.8`, `.6` and `.2`). Therefore 1.7 is not a sum of powers of 2.

answered Oct 21 '10 at 14:59
Benoit
48.3k ●12 ●133 ●190

You can't compare floating point variables for equality. The reason is that decimal fractions are represented as binary ones, that means loss of precision.

answered Oct 21 '10 at 14:59
Andrey
46.6k ●7 ●74 ●129