

Re: double is integer?

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2007-02/msg00497.html>

- *From:* Joseph M. Newcomer <newcomer@xxxxxxxxxxxxx>
 - *Date:* Tue, 06 Feb 2007 10:10:13 -0500
-

Same problem.

In general, it is impossible to deal with the notion that a floating-point value is an "exact" anything, integer or anything else.

```
double d= 10.0;
double q = d / 3.0;
double t = q * 3.0;
```

It is very unlikely that `t == d`, it is equally unlikely that

```
long int i = (long int) d;
if ((double) i == t)
```

will produce anything useful. For example, `i` will be 10, but `(double)i` will be 10.0, and `t` is still 9.999999999999999. Also, why 'long int' instead of just 'long' (the 'int' is a noise word)

It is reasonably safe to assume that any value you have is the result of some computation.

Note that if you don't want users to type in a value that is not an integer, you have to limit what they can type. Or you have to analyze the text of what they type, not the floating-point representation of what they typed.

Floating point arithmetic has a loose grasp of the concept of precision. The fact that its looseness can be rigorously characterized does not make it any less loose.

Any attempts to determine if a floating point value is an exact integer are probably doomed.

For one thing, just the idea of using long is suspect. Is not 5E20 a valid integer? Yes, but it can't be represented in a long, so any transformation that passes through a long is doomed to failure. Note that floating point can represent values that are larger than LONGLONG (__int64). In effect, there is absolutely no way to perform this test reliably.

(If you can accept approximations, then you can format the floating point number and see if there is anything after the decimal point. It still won't work right, but it will be close to right more of the time than comparing the floating point representations. But it

Re: double is integer?

also will fail for all the above reasons, but if you use, say, 3 digits of decimal precision you will only fail some of the time, rather than pretty consistently all the time. And you'll have to deal with the failures, so you haven't solved anything, really)
joe

On Mon, 5 Feb 2007 16:55:23 -0000, "GT" <ContactGT_remove_@xxxxxxxxxxxx> wrote:

"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxx> wrote in message
news:0dcas2lrmirbjg293b14fd9odq39ide0au@xxxxxxxxxxxx

This is problematic at best. While it might work, in general it is not a safe thing to do because what if your value is 2.00000000000001 or 1.99999999999999? After doing a lot of floating point computations, you may have a value close to but not precisely an integer.

For example
double d = 10.0;
double q = d / 3.0;
double t = q * 3.0;

you might assume that $d == t$, which will not be true, and `isInteger(d)` may be true but `isInteger(t)` will not be.

On the whole, this is a very risky thing to consider doing. You should never assume that any computation involving a double will ever produce an integer value. For that matter, if the integer value is very large, the double might lose low-order precision anyway.

There is no reliable way to actually compare doubles to integers and know that the double is an integer value. Typically, you might consider

```
if( (d - ceil(d)) < fuzzfactor)
```

but that is about as good as it gets.
joe

How about:

```
bool isInteger(const double& d)  
{
```

Re: double is integer?

```
long int i = (long int) d;
if ((double) i == d)
{
// we have an integer
return true;
}
else
return false;
}
```

Joseph M. Newcomer [MVP]

email: newcomer@xxxxxxxxxxxxx

Web: <http://www.flounder.com>

MVP Tips: http://www.flounder.com/mvp_tips.htm

.