Subject: Re: About the bits reserved for float variable
Posted by Chris Lee on Fri, 21 May 2004 14:36:01 GMT
View Forum Message <> Reply to Message

In article <c8l0c4jle1@pegasus.fccn.pt>, "Nuno Oliveira"
<nmoliveira@fc.ul.pt> wrote:

> I looking at the Chapter 5 of the Bulding Aplication. It says, for
> float variables that it's a 32 bits number in the range of +/-10^38
> withe approximately six or seven decimal places of significance. What
> I'm missing here? How can a number 32 bit number range between -10^38
> and +10^38?
>

For an 32 bit floating point number, the first bit is the sign bit. the
next 8 bits are the exponent, the last 23 bits are the mantissa (IEEE)

The exponent has 8 bits, it can do -128 -> 128 in base 2,
2^128 = 3.4 x 10^38
2^-128 = ...10^-38

The 23 bits of the mantissa represent a number between 0 and 2 (scaled).
2^23 = 8388608, a 7 digit number

There's an equation to convert them on the IEEE website I think.

Chris.

Subject: Re: About the bits reserved for float variable
Posted by Paul Van Delst[1] on Fri, 21 May 2004 14:44:35 GMT
View Forum Message <> Reply to Message

Nuno Oliveira wrote:
> I looking at the Chapter 5 of the Bulding Aplication.
>
> It says, for float variables that it's a 32 bits number in the range of
> +/-10^38 withe approximately six or seven decimal places of significance.
> What I'm missing here? How can a number 32 bit number range between -10^38
> and +10^38?

Some of the bits are used for the significand, and some of the bits are used for the
exponent. For IEEE 754 arithmetic, a single precision, 32-bit, number uses 23 bits for the
significand (plus one for the sign bit), and 8 for the exponent. With 8 bits for the
exponent, it can range from -127 to 128. 2^-127 ~ 10^-38, 2^128 ~ 10^38.

Similarly for double precision (64 bit) where the significand is 52 bits long and the exponent 11 bits giving a range of ~10^(+/-)308.

Don't quote anything I've said above as being anything other than a 2-bit (ha ha) explanation of a somewhat complicated topic by someone (me) who only understands the very basics.

paulv

Subject: Re: About the bits reserved for float variable
Posted by James Kuyper on Fri, 21 May 2004 15:00:08 GMT
View Forum Message <> Reply to Message

Nuno Oliveira wrote:
> I looking at the Chapter 5 of the Bulding Aplication.
>
> It says, for float variables that it's a 32 bits number in the range of
> +/-10^38 withe approximately six or seven decimal places of significance.
> What I'm missing here? How can a number 32 bit number range between -10^38
> and +10^38?

It can do that by not representing every integer value in that range. A 32-bit type can represent a maximum of 2^32 different values. An ordinary 32 bit integer type represents 2^32 consecutive integer values. A 32-bit IEEE format floating point number represents a slightly smaller set of values (because some of the bit patters represent +infinity, -infinity, denormalized numbers, and NaNs), but those values are very closely spaced near 0, and more widely spaced out the larger the values are, which allows them to cover a much larger dynamic range.

To be specific, an IEEE format number contains a sign bit, a mantissa, an exponent, and has an implicit offset which is used to interpret the value. The value represented by such a number is

(-1)^sign *(1 + mantissa/2^n)*2^(exponent+offset)

where 'n' is the number of bits in the mantissa, and offset is negative. Note that this formula provides no way to represent 0 (the mantissa is never negative). As a special exception, a mantissa and exponent that are both zero are treated as representing 0, rather than 2^offset, which is what the general formula would call for.

Thus, for any given value of 'k' within a certain range, this format can represent exactly 2^n different value x in the range 2^k <= x < 2^k+1, evenly spaced within that interval.

Subject: Re: About the bits reserved for float variable
Posted by David Fanning on Fri, 21 May 2004 15:07:25 GMT
View Forum Message <> Reply to Message

James Kuyper writes:

> It can do that by not representing every integer value in that range. A
> 32-bit type can represent a maximum of 2^32 different values. An
> ordinary 32 bit integer type represents 2^32 consecutive integer values.
> A 32-bit IEEE format floating point number represents a slightly smaller
> set of values (because some of the bit patters represent +infinity,
> -infinity, denormalized numbers, and NaNs), but those values are very
> closely spaced near 0, and more widely spaced out the larger the values
> are, which allows them to cover a much larger dynamic range.
>
> To be specific, an IEEE format number contains a sign bit, a mantissa,
> an exponent, and has an implicit offset which is used to interpret the
> value. The value represented by such a number is
>
> (-1)^sign *(1 + mantissa/2^n)*2^(exponent+offset)
>
> where 'n' is the number of bits in the mantissa, and offset is negative.
> Note that this formula provides no way to represent 0 (the mantissa is
> never negative). As a special exception, a mantissa and exponent that
> are both zero are treated as representing 0, rather than 2^offset, which
> is what the general formula would call for.
>
> Thus, for any given value of 'k' within a certain range, this format can
> represent exactly 2^n different value x in the range 2^k <= x < 2^k+1,
> evenly spaced within that interval.

Nuno, aren't you glad you asked. :-)

This kind of answer has always fallen into the
"Too Much Information" category for me. I think of
it this way, you can have fast or accurate, but you
can't have both. That's about as much as I've ever
needed to know using a computer. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: About the bits reserved for float variable
Posted by Kenneth P. Bowman on Fri, 21 May 2004 22:22:01 GMT
View Forum Message <> Reply to Message

In article <MPG.1b17c91c4c958ba1989761@news.frii.com>,
David Fanning <davidf@dfanning.com> wrote:

> it this way, you can have fast or accurate, but you
> can't have both. That's about as much as I've ever
> needed to know using a computer. :-)

It used to be that integer arithmetic was faster than floating point, but that is generally no longer the case. Just about all machines that I know of can do integer or floating point ops in a single clock cycle. Some cpus can do more than one op per clock cycle. (That's what many of those millions of transistors on modern cpus are used for.)

Additionally, many (but not all) architectures have double-precision floating point hardware units. DP operations on those systems are as fast as single precision. On many machines the only drawbacks to doing everything in DP are: twice as much memory is required and twice as much file space.

My rules of thumb:
 Use integers for things you can count (i.e., no fractions).
 Use doubles for "real numbers", unless memory is a problem.
 Write files in single or double precision, as needed.

Ken Bowman

Subject: Re: About the bits reserved for float variable
Posted by George N. White III on Sat, 22 May 2004 12:01:42 GMT
View Forum Message <> Reply to Message

On Fri, 21 May 2004, Nuno Oliveira wrote:

> I looking at the Chapter 5 of the Bulding Aplication.
>
> It says, for float variables that it's a 32 bits number in the range of
> +/-10^38 withe approximately six or seven decimal places of significance.
> What I'm missing here? How can a number 32 bit number range between -10^38
> and +10^38?

Read chapter 1 of any decent introductory numerical analysis text and then "What Every Computer Scientist Should Know About Floating-Point Arithmetic" by D. Goldberg, ACM Computing Surveys, Vol 23, 1991, p5-48) and reprinted in Sun's online manuals. See links on:

    http://cch.loria.fr/documentation/IEEE754/

While it is true that many people do get by without understanding
f.p. arithmetic, there are also many examples of calculations going astray
due to failure to recognize situations where the difference between
f.p. and real numbers matters. It is becoming more important to
understand the material in Goldberg's paper because newer hardware
speedups (speculative execution, merged operations, parallel
processing) tend to make it harder to diagnose arithmetic problems.

--
George N. White III <aa056@chebucto.ns.ca>

Subject: Re: About the bits reserved for float variable
Posted by Nuno Oliveira on Tue, 25 May 2004 17:08:29 GMT
View Forum Message <> Reply to Message

Those were definitely careful answers. Indeed, more that I needed to know.
But anyway I enjoyed knowing all those things. And besides that it's a great
thing to know that are that like to help the others. Enchanted!, I can say.

"David Fanning" <davidf@dfanning.com> wrote in message
news:MPG.1b17c91c4c958ba1989761@news.frii.com...
>
> Nuno, aren't you glad you asked. :-)
>
> This kind of answer has always fallen into the
> "Too Much Information" category for me. I think of
> it this way, you can have fast or accurate, but you
> can't have both. That's about as much as I've ever
> needed to know using a computer. :-)
>
> Cheers,
>
> David
>
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/