

Re: A Simplified Number System

Source: <http://sci.tech-archive.net/Archive/sci.lang/2004-11/0344.html>

From: Sean O'Leathlobhair (jwlawler_at_yahoo.com)

Date: 11/03/04

Date: 3 Nov 2004 01:42:27 -0800

Tak To <takto@alum.mit.edu.-> wrote in message news:<lpadnSCueY3LjhXcRVn-sQ@comcast.com>...

> Sean wrote:

>> *What are your "fixed point numbers"? The ones most familiar*

>> *to me are BCD.*

>

> Lee Sau Dan wrote:

>> *Fixed point is as opposed to floating point. Recall that a floating*

>> *point number consists of 2 (signed) parts: mantissa and exponent.*

>> *With fixed point, I mean the exponent is always fixed. So, you only*

>> *need to store the mantissa. The exponent is implicit. You would of*

>> *course use exponents that are (negative) powers of 10 for financial*

>> *applications. A reasonable way to represent the mantissa in binary is*

>> *to use 2's complement form.*

>

> Wiktor S. wrote:

>> *It's like keeping all prices in cents instead of dollars and it's*

>> *fractions (or whatever currencies you use). I think it's the best*

>> *solution. Normal integers and normal arithmetics. You only need to*

>> *display them correctly. No weird BCD, no special structures.*

>

> *In some programming languages such as PL/1, one can declare fixed*

> *point numbers as, for example, FIXED(5,2), meaning a precision of*

> *5 (decimal) digits for the integral part, 2 digits for the fraction*

> *part. The compiler would then do all the conversion for you.*

>

> Tak

An even more important example is SQL which is the dominant database language. In a similar way you may declare types such as DECIMAL(11,2).

The implementer of a particular database is free to choose any technique provided it functions correctly. A compliant product could be built using scaled integers but I am unaware of one. But when the hardware has BCD support then this is a good choice and this is very popular.

sci.lang: Re: A Simplified Number System

What's wrong with BCD? Those familiar with it think it is the obvious easy solution and most like the way humans do arithmetic (*). When introducing people with a Cobol background to a language such as C++, I have to spend quite a bit of time explaining the weird types int and double.

(*) This is very important in financial applications. Auditors do things in a particular human way and will call anything else "wrong". It is no good explaining the intricacies of computer arithmetic to them.

What you are used to seems normal and sensible and what you are unused to seems weird and unnecessary. It works both ways around.

Seán O'Leathlóbhair