

The background features a series of vertical lines in various colors (green, blue, purple, red, yellow) that create a sense of depth and movement. On the right side, there is a 3D wireframe structure composed of many thin lines, resembling a corner or a perspective view of a grid. The overall color palette is dark, with the lines providing a vibrant contrast.

データ型の 誤差について

C言語の型の表せる範囲

実行環境による

sizeof演算子で調べることが可能
この演算子は、バイト数を返す

- | | | | |
|-----------------|-----|---------------|-----|
| • int | : 4 | • long | : 8 |
| • unsigned int | : 4 | • long long | : 8 |
| • long int | : 8 | • float | : 4 |
| • long long int | : 8 | • double | : 8 |
| • char | : 1 | • long double | : 8 |
| • unsigned char | : 1 | | |

2進数で小数を表す

$$(10)_{10} = (1010)_2$$

$$(0.5)_{10} = (0000.1000)_2$$

$$(1.75)_{10} = (0001.1100)_2$$

$$(0.1)_{10} = (?)_2$$

0.1は表すことができるだろうか

2進数の桁	1	1	1	1	.	1	1	1	1	1	1	1	1
10進法	8	4	2	1		0.5	0.25	0.125	0.0625	0.0312 5	0.0156 25	0.0078 125	0.0039 0625

0.1を表せない計算機

C言語では、標準のライブラリでは、0.1を正確に扱うことができず、丸め誤差ができる。

これは、計算結果が厳密な値ではなく、わずかに異なる数値になる場合があるということ。

特に繰り返し処理で丸め誤差を意図的に蓄積させると、その差が顕著に出る。

	0.01を10,000回加算	0.01を10,000,000回加算
float	100.002953	95680.945312
double	100.000000	99999.999986
正しい値	100.000000	100000.000000

小数を整数として扱う

計算機たるものが、誤差があってはいけない。

実際、プログラムでは誤差を考えて作られている場合が多くある。

誤差が許されない場合では、小数を整数として扱ったりなど、さまざまな方法で小数を扱って計算している。

以下に小数を整数として扱う型（decimal）を自作した。その結果を記す。

	0.01を10,000回加算	0.01を10,000,000回加算
float	100.002953	95680.945312
double	100.000000	99999.999986
decimal	100.000000	100000.000000
正しい値	100.000000	100000.000000

大きな型

先ほどのdecimalで扱えるデータの大きさは、16バイトにしている。

小数に限らず、大きな桁数の値を扱う際は、さらに大きなデータ型を用意する必要がある。

筆者も4096bit（512バイト）のint型を作成したことがある。

リンク：<https://slrte.com/slrte/investigation/CDR/int4096.html>

	0.01を10,000回加算	0.01を10,000,000回加算
float	100.002953	95680.945312
double	100.000000	99999.999986
decimal	100.000000	100000.000000
正しい値	100.000000	100000.000000
