

2進数⇔10進数変換

2進数"11.0010001"を10進数に変換する

0	0	1	1	.	0	0	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---

※ Binary Code decimal、通常、小数点を含む大きな数の計算はFloat浮動小数点表示、double倍精度で表現される。

$$\begin{aligned} & 0 \times (2^3) + 0 \times (2^2) + 1 \times (2^1) + 1 \times (2^0) + 0 \times (2^{-1}) + 0 \times (2^{-2}) + 1 \times (2^{-3}) + \\ & 0 \times (2^{-4}) + 0 \times (2^{-5}) + 1 \times (2^{-6}) \\ & = 0 \times 8 + 0 \times 4 + 1 \times 2 + 1 \times 1 + 0 \times 1/2 + 0 \times 1/4 + 1 \times 1/8 + 0 \times 1/16 + 0 \times 1/32 + 1 \times 1/64 \\ & = 0 + 0 + 2 + 1 + 0 + 0 + 0.125 + 0 + 0 + 0.015625 \\ & = 3.140625 \quad (\approx 3.14) \end{aligned}$$

10進数3.141593を2進数に変換する

整数部 3 と小数部 0.141593 (6桁)に分けて計算する。

整数→商がゼロになるまで2で割る、余りを下から並べる。

$$3 \div 2 = 1 \dots 1$$

$$1 \div 2 = 0 \dots 1$$

$$1 \div 2 = 0 \dots 1 \quad \uparrow \text{こちらからbitを並べる、結果：11}$$

小数→積の小数がゼロになるまで2を掛ける（積の小数だけを引き継ぐ）

$$0.141593 \times 2 = 0.283186 \quad \downarrow$$

$$0.283186 \times 2 = 0.566372$$

$$0.566372 \times 2 = 1.132744$$

$$0.132744 \times 2 = 0.265488$$

$$0.265488 \times 2 = 0.530976$$

$$0.530976 \times 2 = 1.061952$$

.

.

積の整数を上からbitを並べる、結果：001001 したがって、11.001001

2進数表示の円周率πの桁数をさらに増やすと、

11.001001000011111101101010100010001000010110100011

C++でのプログラミング例

2進数のデータの型には、整数型 (int)、浮動小数点型 (float、double、long double)10進型 (decimal, BCD)などがある。

```
iPad 4:17 AM
New Open Send Undo Redo Build Run
1 #include <iostream>
2 #include <math.h>
3 using namespace std;
4 int main() {
5
6     long double fraDecimal, fraBinary, bFractional = 0.0, dFractional,
    fraFactor=0.1;
7     long int dIntegral, bIntegral=0;
8     long int intFactor=1, remainder,temp, i;
9
10    cout << "Hello, c++ World!" << endl;
11    cin >> fraDecimal;
12    cout << "Enter any fractional decimal number:" << fraDecimal << endl;
13    cin >> fraDecimal;
14    dIntegral = fraDecimal;
15    dFractional = fraDecimal - dIntegral;
16
17    while(dIntegral!=0){
18        remainder=dIntegral%2;
19        bIntegral=bIntegral+remainder*intFactor;
20        dIntegral=dIntegral/2;
21        intFactor=intFactor*10;    }
22
23    for(i=1;i<=6;i++){
24        dFractional = dFractional * 2;
25        temp = dFractional;
26        bFractional = bFractional + fraFactor* temp;
27        if(temp ==1)
28            dFractional = dFractional - temp;
29
30        fraFactor=fraFactor/10;
31    }
32
33    fraBinary = bIntegral + bFractional;
34    cout << "Equivalent binary value:" << fraBinary;
35
36 }
```

```
Enter any fractional decimal number: 3.141593
Equivalent binary value: 11.001001
```

演算誤差

コンピュータの中で小数計算をしていて、結果が正しくないときがある。
理由は、ほとんどの10進数の小数は有限桁の2進数で表現することができないことによる。

SingleやDoubleのような浮動小数点数型(Float)は、値を2進数で格納している。
その近似法は、IEEE754で定義されている。

- ・ 最近接偶数への丸め
- ・ 0への丸め
- ・ 正の無限大への丸め
- ・ 負の無限大への丸め

浮動小数点数型では、近似値でしか表現することができず、その誤差が上記のような非常識的な計算結果として現れる。

例えば、10進数の0.1を2進数に変換すると”0.0001100110011...”となり、0011の部分が循環する。

よって、0.1をSingleやDouble型に格納するには、適当な桁で近似する必要がある。

ここでの丸め方は、最近接偶数への丸めを使う。

結果として「0.1」はDouble型では2進数で”

“0.000110011001100110011001100110011001100110011001100110011001101”

これを10進数に戻すと、

0.1000000000000000055511151231257827021181583404541015625となり、0.1ではない。

データ型にはBinary Decimal (Decimal型)もあり、この型では小数のすべての桁をbyteであらわすので、近似の必要ないが、科学計算向きではない。

※ 同じ数値を2回以上丸めてはいけない。たとえば、122.51を最近接偶数へ丸めるときに、まず122.5とし、次に122とすると、結果が違ってしまう。

Decimal型での計算結果は、ソロバン電卓と同じになる。

例えば、 $1 \div 3 \times 3 = 0.9999999\dots$

しかし、数学では、 $0.9999999\dots = 1$ と読む。

