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.

0×(2^3) + 0×(2^2) + 1×(2^1) + 1×(2^0) . 0×(2^-1) + 0×(2^-2) + 1×(2^-3) +
0×(2^-4) + 0×(2^-5) + 1×(2^-6)
= 0×8 + 0×4 + 1×2 + 1×1 . 0×1/2 + 0×1/4 + 1×1/8 + 0x1/16 + 0x1/32 + 1x1/64
= 0 + 0 + 2 + 1 . 0 + 0 + 0.125 + 0 + 0 + 0.015625
= 3.140625 (≒ 3.14)

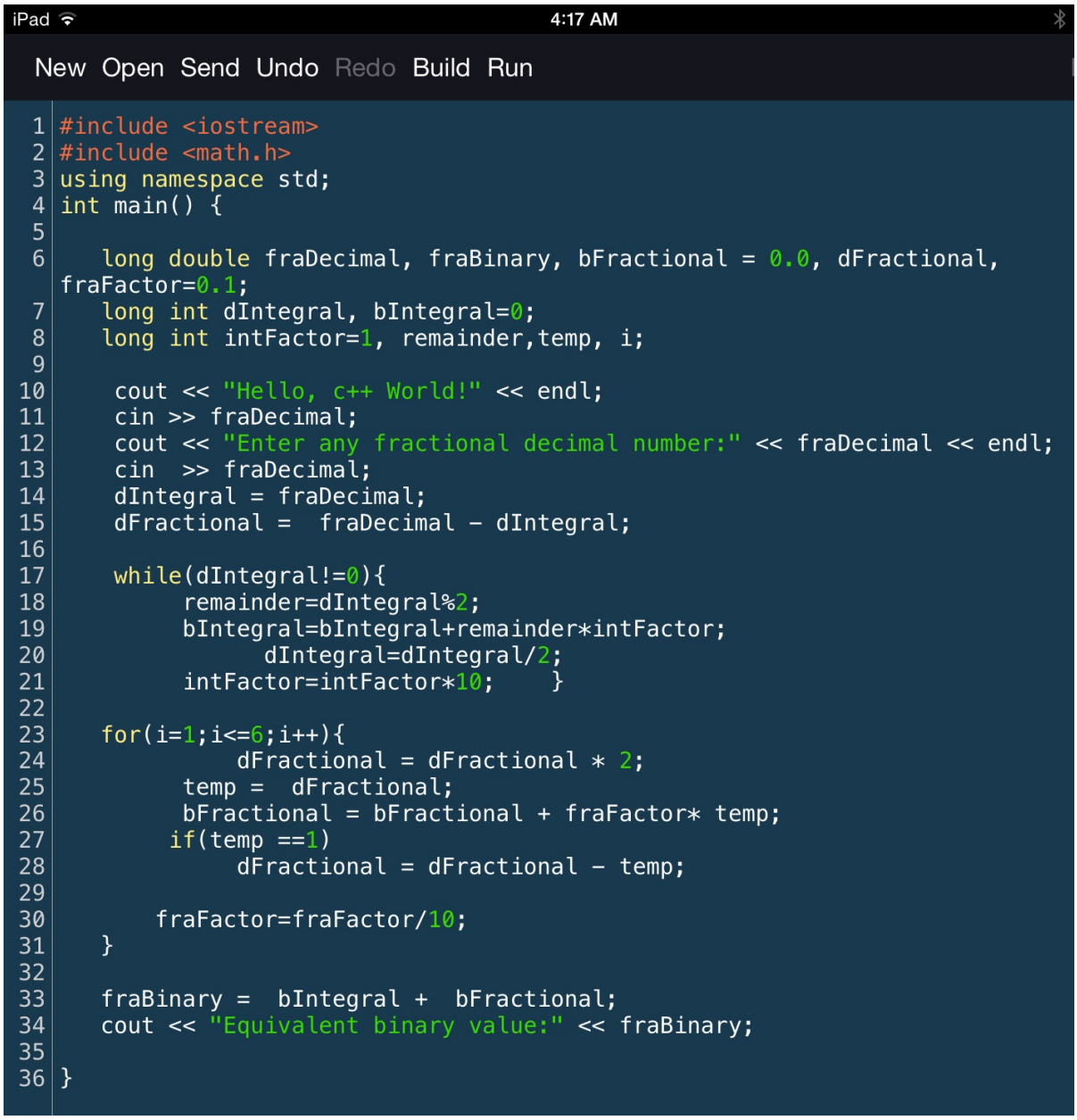
10進数3.141593を2進数に変換する

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 ÷ 2 = 1 . . . 1
1 ÷ 2 = 0 . . . 1
1 ÷ 2 = 0 . . . 1　↑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 × 2 = 0.283186 ↓
0.283186 x 2 = 0.566372
0.566372 x 2 = 1.132744
0.132744 x 2 = 0.265488
0.265488 x 2 = 0.530976
0.530976 x 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、
11.00100100001111111011010101000100010000101101000 11 ....

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.




```
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”
“0.0001100110011001100110011001100110011001100110011001101”

これを10進数に戻すと、
0.1000000000000000055511151231257827021181583404541015625となり、0.1ではない。

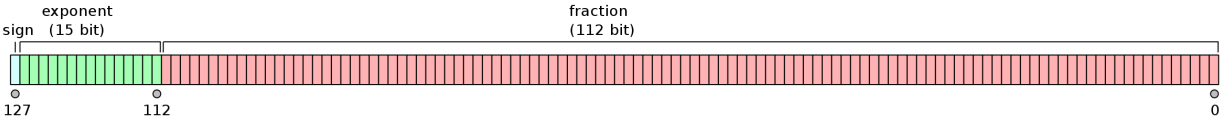
データ型にはBinary Decimal (Decimal型)もあり、この型では小数のすべての桁をbyteであらわすので、近似の必要ないが、科学計算向きではない。

※　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➗3✖3 = 0.9999999….
しかし、数学では、0.9999999….. = 1 と読む。


符号
指数部
仮数部


sign
exponent
(15 bit)
fraction
(112 bit)
127
112
0