第1章 CPUが実数を扱う方法を理解して正確な数値演算プログラムを書く

浮動小数点数演算の基礎知識

高田 昌之

リスト1　0.1を積算するプログラム

```
#include <stdio.h>

int main() {
  float a;
  a = 0;
  while ( a != 0.3 ) {
    a += 0.1;
    printf( "a: %g\n", a );
  }
}
```

リスト2　Cコンパイラclangの出力メッセージ

```
takata@Air24:~/FPN[1161] cc -O0 -o test1 test1.c
test1.c:7:15: warning: floating-point comparison is always true; constant
          cannot be represented exactly in type 'float' [-Wliteral-range]
    7 |     while ( a != 0.3 ) {
      |             ~ ^  ~~~
1 warning generated.
takata@Air24:~/FPN[1162] cc --version
```

プログラムの変数には型があり，扱いたいデータに応じて型を決めます．しかしデータだけを見ていると型の本質を見誤るかもしれません．型はCPUに対して，どのような演算を行うか指令するものです．

CPUは演算用の回路をたくさん持っており，オペランドの型によって使用する回路が変わります．オブジェクト・コード（実行可能コード）を見ると，使用する型に応じたCPU命令が並んでいます．

● 浮動小数点数の扱いは抽象化しきれない

CPUがどの回路を使うかはプログラミング言語によって抽象化され，ソフトウェアのレイヤからは見えづらいですが，抽象化しきれない部分もあります．その代表例として浮動小数点数の扱いが挙げられます．大きな数と小さな数の計算では，必要な精度を維持するために浮動小数点数を正しく理解して選択する必要があります．数値演算の最適化をする場合にも，使用するCPUのアーキテクチャにおける浮動小数点数の扱いを知っておく必要があります．

本稿ではCPU上で浮動小数点数がどのように扱われるかを解説します．

浮動小数点数の不思議な振る舞い

浮動小数点数がコンピュータ上でどのように扱われるかを知っておかないと，正しい計算結果を得られない場合があります．そのような振る舞いを確認するため**リスト1**のプログラムを実行します．

このプログラムは，変数aの値を0から0.1ずつ増やし，0.3になるまでwhileループを繰り返してaの値を画面に出力します．

これを筆者の環境のCコンパイラ[注1]でコンパイルすると，**リスト2**のメッセージが表示されました．

「浮動小数点数の比較は常に真になります；定数はfloat型では正確には表現不可能です」という警告メッセージです．エラー・メッセージではないので**リスト3**のように実行できます．0.3で止まらず無限ループになっています．

変数aの型をfloatからdoubleに変えると，コンパイル時には何も指摘されませんが，実行結果はfloatのときと同じです．

次に変数の値に0.01ずつ加算する**リスト4**のプログラムを実行してみます．

このプログラムでは変数aの値を0から0.1まで0.01間隔で増やします．Cのプログラムを見ると，aが0.1になればループ内の処理は実行されなさそうです．しかし，**リスト5**の通り実行結果を見ると0.11が出力されています．

このような挙動は，浮動小数点数の表現の方法に起因して起こります．

整数と浮動小数点数の表現方法

● 整数表現に対するC言語の要求仕様

コンピュータは2進数で全ての情報を記述します．C言語がこれをどう扱うかの規定はいろいろありますが，筆者の手元にあったISO/IEC 9899:1999の規格を

注1：macOS 14.7.5上でclang-1600.0.26.6を使用しました．