Cプログラミング1(再)
第4回

講義では、Cプログラミングの基本を学び
演習では、やや実践的なプログラミングを通して学ぶ

コンピュータにおけるデータの表現方法

コンピュータの内部は２進数で表現されている

基本単位はビット (bit)

　　1ビット　は 一つの ON/OFF 状態の表現

　　もしくは　0 か 1 という状態を表す

コンピュータの処理は「バイト(byte)」単位で処理

　　1 byte = 8 bits

数の表現方法

デジタル (digital)

デジタル	vs	アナログ
離散、不連続		連続

昔は「アナログ型コンピュータ」というものがあった
今のコンピュータは「デジタル型コンピュータ」
　すべての物事を「デジタル (= 飛び飛びの値)」として表現
　精度を高めることで問題を解決

コンピュータは有限の桁数を処理

50の階乗を求めよ

(1) C でプログラムする…

(2) Python で次を実行する:

```
fact = 1 ;  n=50
for  i  in  range(1,n+1) :
        fact *=i
 print(n,"! =", fact)
```

```
int main(void) {
    int i,fact=1,n=50;
   for(i=1; i<=n; i++) {
        fact *= i;
}
    printf("%d! = %d¥n", n, fact);
    return 0;
}
```

Cには「型」がある。それぞれの型により表現できる範囲が決まっている

データには型がある

Cの変数で記憶できるもの(データ)は基本的に

数

である

それは整数(integer)と浮動小点数(floating point number)の2種類

疑問: 文字や配列（文字列を含む）はどうなの？

ひとまずの解答: 文字は「符号なしの整数」
配列は「ポインタ」（記憶番地）でこれも内部的には符号なしの整数

いろいろな整数の表記法

- 符号なし整数
  例: 5U

- 10進数
  数字の並び　　例: 123

- 16進数
  先頭に 0x または 0X をつける　例: 0xc8

- 8進数
  先頭に0をつける　　例: 0310

いろいろな数を表示してみよう

printf の第1引数の書式指定において

%d と書くと、対応する整数を10進表示

したが、8進や16進で表示するのはどうするのか？

答: %o とすると 8進表示、 %x で16進表示

```
#include <stdio.h>

int main(void) {
   int i;
   for(i=1; i<=100; i++)
        printf("%5d  %5o  %5x¥n", i, i, i);
  return 0;
}
```

```
    1    1    1
    2    2    2
    3    3    3
    4    4    4
    5    5    5
    6    6    6
    7    7    7
    8   10    8
    9   11    9
   10   12    a
   11   13    b
   12   14    c
   13   15    d
   14   16    e
   15   17    f
   16   20   10
   17   21   11
   18   22   12
   19   23   13
   20   24   14
   21   25   15
   22   26   16
   23   27   17
   24   30   18
   25   31   19
   26   32   1a
```

オーバーフローについて

• オーバーフロー(overflow): 桁あふれ

　　Cの数として表現できるものには限界がある

⇒　表現できる範囲を超えると正常な結果が得られない

Cでは、整数を「2進数」として内部表現していました。

　int型の変数で記憶できるのは32ビット＝－2^{31} ~ 2^{31}-1までの数を表現

-2147483648　　2147483647≒2.15*10^9

　参考: 13! ≒ 6.227 *10^9

例：階乗計算プログラムで

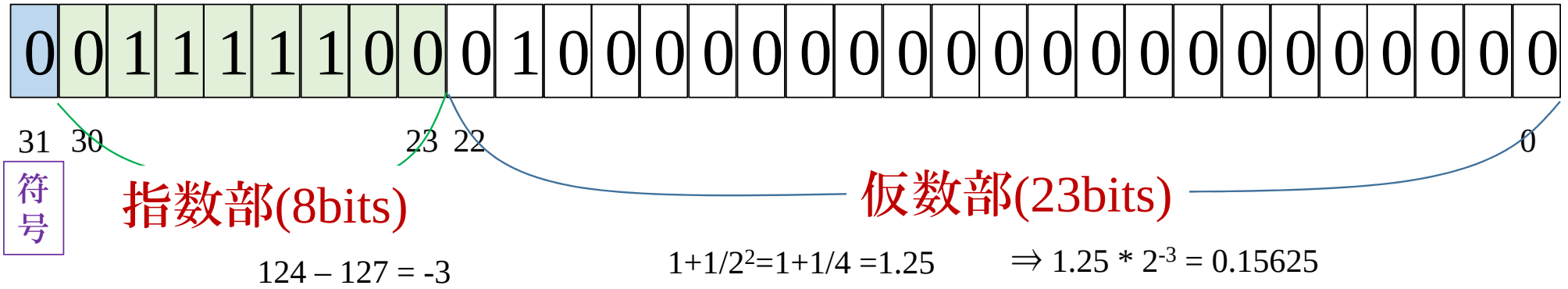
　10から20までの階乗計算を表示させてみよう

　(ウェブ上の参考プログラム「階乗計算」参照)

　　http://lang.sist.chukyo-u.ac.jp/Classes/C/fact.c

浮動小数点数

Cではfloat型の数は次のような形で表現される(0.15625を表す):



おおまかに言って、指数部で2の乗数を、仮数部で小数点以下の数（精度）を表す
これで表現できる正の数のは、$2^{-149} \fallingdotseq 1.4 \times 10^{-45}$ であり、
　それよりも小さな数は0 として扱われる --- アンダーフロー(undeflow)

したがって、精度を気にしなければ、大きな数や小さな数は「対数」計算を用いる

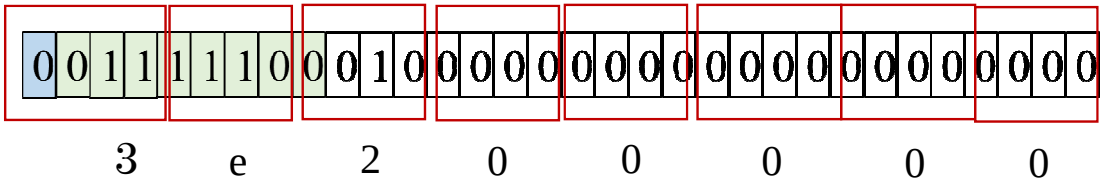
小数点数の内部表現の表示プログラム

浮動小数点数を入力し、その内部表現を16進数として表示する
(warningは出るが動くはず)

```
#include <stdio.h>

int main(void) {
        int i;
        printf(“小数点数？ >");
        scanf("%f", &i);
        printf("%x¥n", i);
  return 0;
}
```

```
C:¥Users¥sirai¥C>float.exe
小数点数？ >0.15625
3e200000
```



文字の表現方法

ASCII文字セット（ウェブ検索して調べてみよう）

(半角)アルファベットや数字、スペースなどの記号（キーボードに刻印されている日本語文字以外の文字）は1バイトで表現

‘A’ は 数としては 0x41　　‘a’は 0x61

‘B’ は 数としては 0x42　　‘b’は 0x62

…

‘0’は数としては 0x30

‘1’は数としては 0x31

…

いろいろな文字と数の関係を表示する

```
#include <stdio.h>

int main(void) {
   char c;
   for(c='A'; c<='Z'; c++)
         printf("%c:¥t%5d  %5o  %5x¥n", c, c, c, c);
   printf("------------¥n");
   for(c='a'; c<='z'; c++)
         printf("%c:¥t%5d  %5o  %5x¥n", c, c, c, c);
  return 0;
}
```

出力結果：

```
A:        65    101     41
B:        66    102     42
C:        67    103     43
D:        68    104     44
E:        69    105     45
F:        70    106     46
G:        71    107     47
H:        72    110     48
（中途略）
------------
a:        97    141     61
b:        98    142     62
c:        99    143     63
d:       100    144     64
e:       101    145     65
f:       102    146     66
（以下略）
```

制御コードとエスケープ・シーケンス

制御コード

文字、数字、記号のようには『表示されない』モノ

例：改行、タブなど

これらを「表現する」にはどうするか？

よくある方法は、¥n ¥t ¥r ¥f のように、

「¥とアルファベットの組み合わせ」を用いて表す

¥は(環境によっては＼)「次の文字を特別扱いせよ」を表す

⇒ エスケープ文字

日本語の文字表現

アルファベットや数字、記号は 1バイトで表現可能(100個もないため）

それに対して日本語の文字は数千個(常用漢字で2,136）

⇒　表現するには少なくとも2バイト必要

　加えてASCII コードとの両立が望ましい

JISコード(ISO-2022-JP)--- 通常、¥e $ B と ¥e (B というエスケープシーケンスではさまれた「2バイトずつのコード」

Shift-JISコード、EUCコード

Unicode --- UTF-8, UTF-16

宿題

(1) 整数を入力し、その8進数と16進数それぞれの表現を返すプログラムを書いてみよう

実行例：青字は入力

数？> 123

8進数:　173　16進数: 7b

(2) 16進数を入力し、その10進表現を返すプログラムを書こう

実行例：青字は入力

16進数？> abcd

10進数：43981