プログラミングI 第2回
数理物理，総合理学等向け

2018年10月1日

海谷 治彦

目次

- 第2章 (レ) 2章(明)「数値の計算」と
- 3章(レ) 1-2章(明)「変数」の内容となります.

- 整数における加減乗除の計算
- その結果をprintfで表示する方法
- 数学の変数との違い
- 変数を定義(宣言)する
- 型について
- 変数に値をしまう
- 変数の値を読む
- 実数の表示フォーマット (printf)
- ユーザーが入力した数値を変数に読み込む

四則演算の記号がちょっと違う

- 足し算 ＋ ⇒ +
  - 例　3+2
- 引き算 ー ⇒ -
  - 例　5-8
- 掛け算 × ⇒ *
  - 3*4
  - アスタリスクと呼ばれる記号
- 割り算 ÷ ⇒ /
  - 7/3

演算子の結合度

- どの演算子がくっつき具合が強いかは，(たぶん) 通常の数学や算数と同じ

```
    123 +  45 * 67  -  8 / 2
  = 123 + (45 * 67) - (8 / 2)
```

- すなわち * / のほうを - + よりも先に計算する.
- ( ) がついた場合，( ) の内側から計算する.

計算結果の表示

- 計算結果を表示するのにも，printf を用いることができる.
- そもそも，printf は，

  Print format

の略語，数値や文字列等のデータを整形(format)して印字(print)する.

- Decimal 十進法数のこと

List 2-1(レ) 改 p.4-7 (明)

```
01: #include <stdio.h>
02:
03:
04:
05: int main(void)
06: {
07:     printf(“Kasan %d desu.¥n”, 3+2);
08:     printf(“Genzan %d desu.¥n”, 5-8);
09:     printf(“jouzan %d desu.¥n”, 3*4);
10:     printf(“Jozan %d desu.¥n”, 7/3);
11:     return(0);
12: }
```

※ 注意！ 行番号がついたままでは，コンパイルできません．

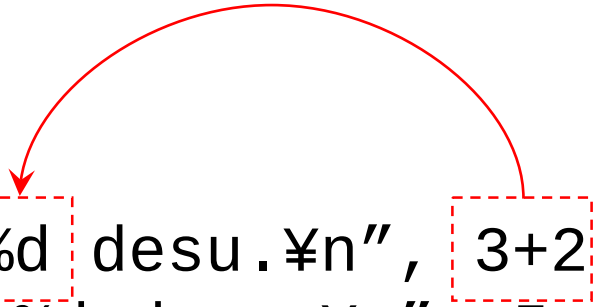
結果は以下になる

```
01: #include <stdio.h>
02:
03:
04:
05: int main(void)
06: {
07:     printf(“Kasan %d desu.¥n”, 3+2);
08:     printf(“Genzan %d desu.¥n”, 5-8);
09:     printf(“jouzan %d desu.¥n”, 3*4);
10:     printf(“Jozan %d desu.¥n”, 7/3);
11:     return(0);
12: }
```

```
Kasan 5 desu.
Genzan -3 desu.
jouzan 12 desu.
Jozan 2 desu.
```

%d に結果が埋め込まれる

```
01: #include <stdio.h>
02:
03:
04:
05: int main(void)
06: {
07:     printf(“Kasan %d desu.¥n”, 3+2);
08:     printf(“Genzan %d desu.¥n”, 5-8);
09:     printf(“jouzan %d desu.¥n”, 3*4);
10:     printf(“Jozan %d desu.¥n”, 7/3);
11:     return(0);
12: }
```



割り算では結果は切り捨て

```
01: #include <stdio.h>
02:
03:
04:
05: int main(void)
06: {
07:     printf(“Kasan %d desu.¥n”, 3+2);
08:     printf(“Genzan %d desu.¥n”, 5-8);
09:     printf(“jouzan %d desu.¥n”, 3*4);
10:     printf(“Jozan %d desu.¥n”, 8/3);
11:     return(0);
12: }
```

```
Kasan 5 desu.
Genzan -3 desu.
jouzan 12 desu.
Jozan 2 desu.
```

8/3は2.6･･･であるが，3ではなく2

%d は複数回使える (fig. 2-4類似)

ソースプログラム

```
#include <stdio.h>

main(){
 printf("tasu %d hiku %d kake %d wari %d¥n", 3+2, 5-8, 3*4, 8/3);
}
```

結果

```
tasu 5 hiku -3 kake 12 wari 2
```

表示ケタ数の指定

- 数値の桁数の幅を指定したい場合，%5d 等で，5 ケタ等が指定できる.

```
// keta.c
#include <stdio.h>

main(){
 printf("%8d miso¥n", 340);
 printf("%8d sato¥n", 45);
 printf("%8d wine¥n", 4321);
}
```

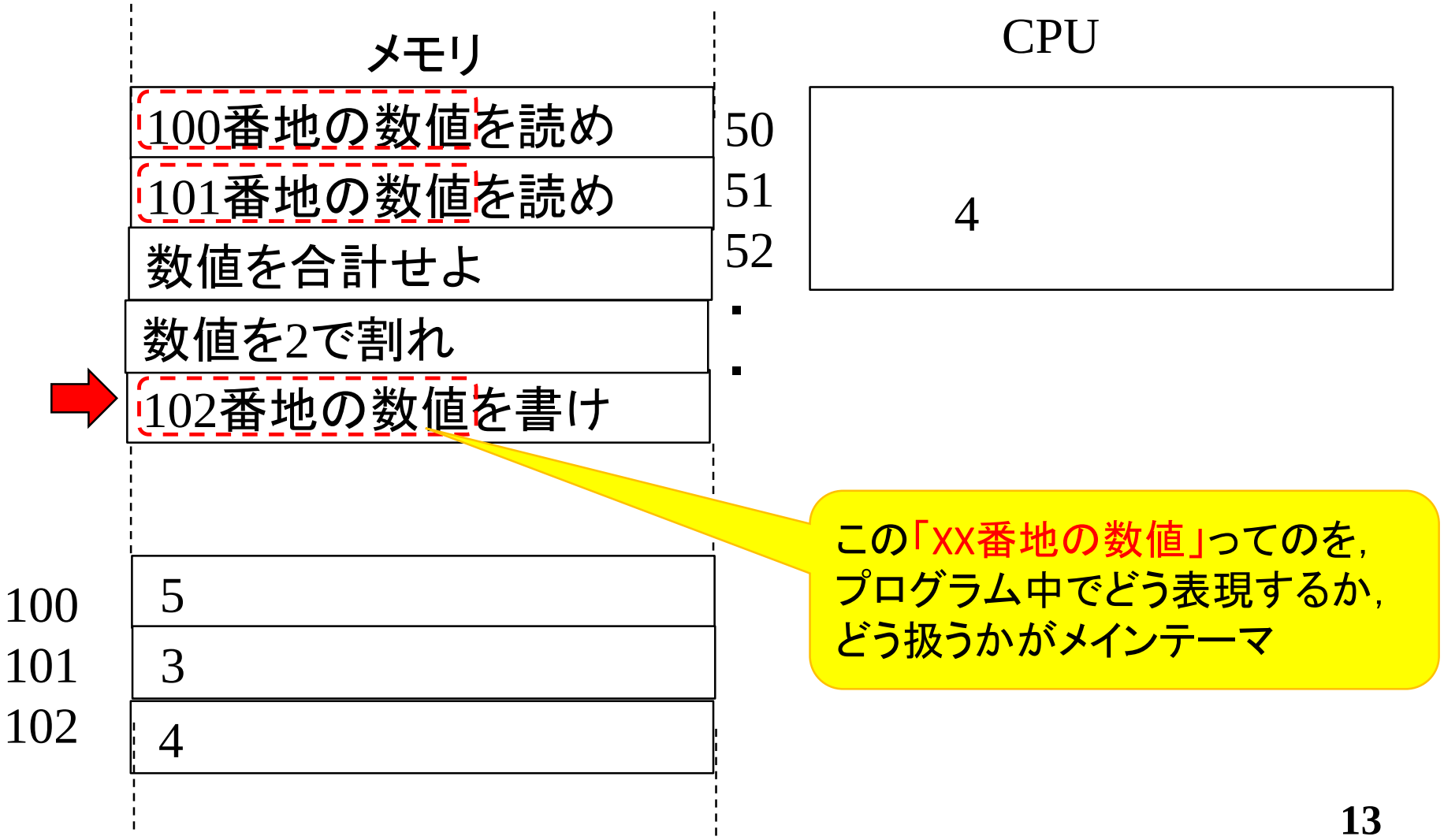
```
     340 miso
      45 sato
    4321 wine
```

変数について

とりあえず数値のみ

復習

簡易な例題 ～ 二値の平均



数学の変数とは違う

- 数学の変数
  - 方程式の中の変数は，同じシンボル(例えば x, y 等)なら，同じ値を持つものとみなす．

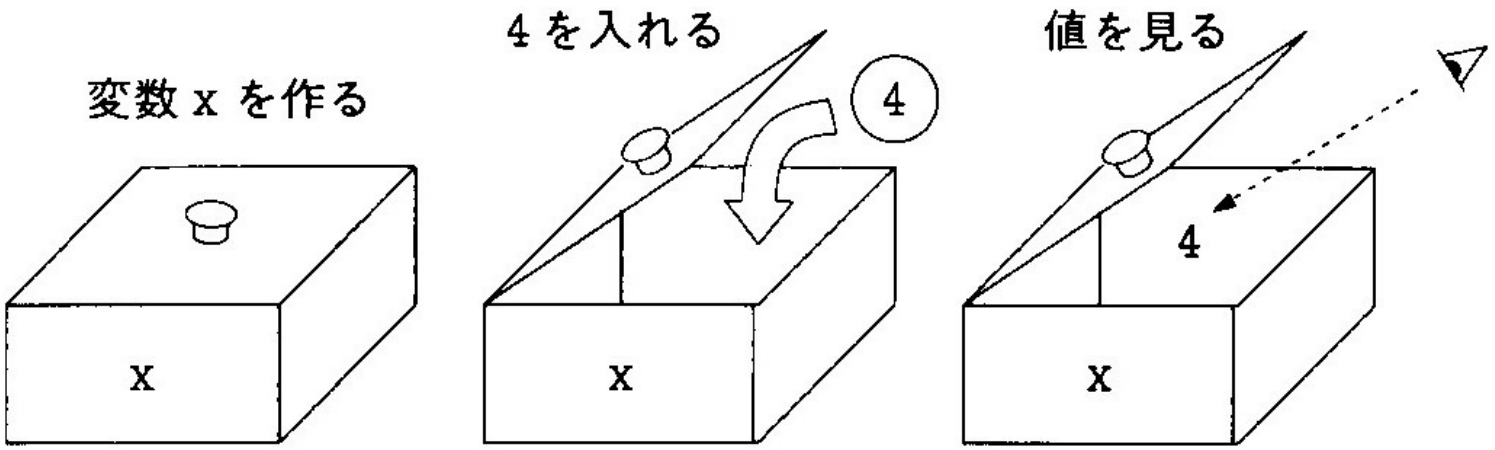
  例　$x^2+2=3x$

  x は 1 もしくは 2 どちらかの値をとる．
  - 定義式の変数も同様である．
  - 例 $f(x)=x^2-3x+2$
    - f(1)=0, f(2)=0, f(3)=2 等
  - 要は未知の値に便宜上，名前をつけているモノである．
- Cプログラムの変数
  - 変数は値の保管場所の名前である．
  - 保管してある値は，ある瞬間には1個である．
  - 同じ変数でも，実際の値はプログラムによって変わる．

C言語の変数と基本操作

- 変数は値の保管箱なので，以下の三種類の操作を行うことができる．
  - 変数定義: 値を保管する場所の確保
    - Declaration の訳語なので，定義より宣言のほうが一般的．
  - 代入: 値を保管する
  - 参照: 保管してある値を見る，使う
- 以降，それぞれについて解説する，その前に

p.10-13
(明)



変数の型について

- 値の種類によって保管箱の大きさが異なる.
  - 単なる整数，例えば 124 よりも，実数，例えば 3.14159254 を保管する箱のほうが大きめじゃないと困るでしょ.
- よって，型というもので変数の種類(保管場所の大きさ)を区別する.
- C言語には多様な型があるが，当面，以下くらいを知っていればよい.
  - `int` 整数を保管する変数の型，英単語integerに由来.
  - `float` 実数を保管する箱の型

    (通常 int と同じサイズなので精度が悪い，浮動小数点と呼ばれる)
  - `double` 倍精度実数を保管する型
  - `char` 文字1個を保管する型，英単語characterに由来.

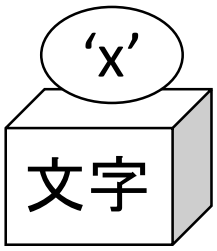
p.28-31
(明)

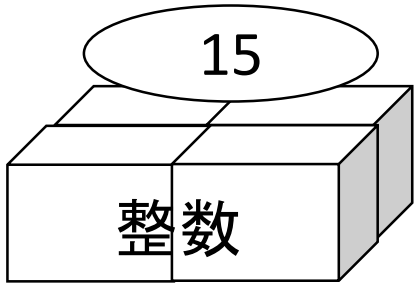
参考 箱の大きさ

- 型によって一個の変数の大きさは異なる.
  - char を1箱とすると,
  - int 4箱分
  - float 4箱分
  - double 8箱分

が一般的. しかし, C言語にはこの辺の明確な規定が無い.

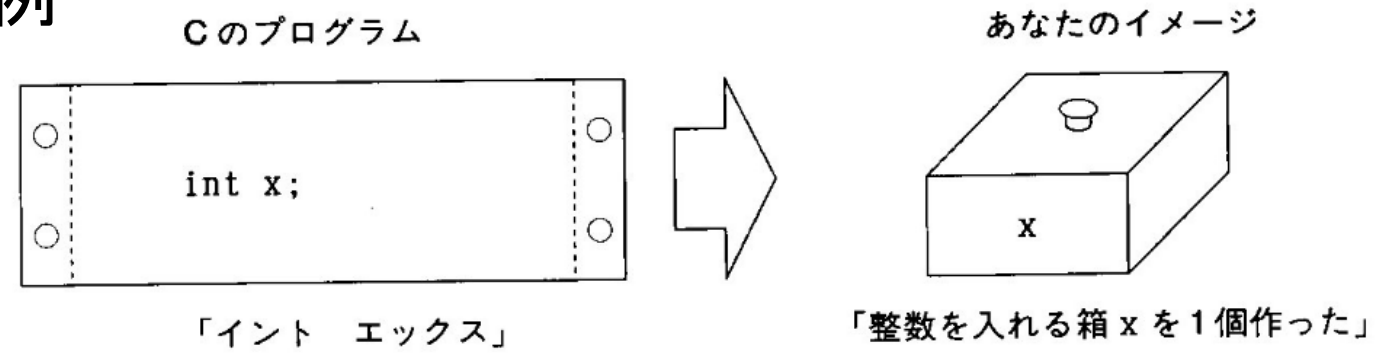
- ハードウェアの効率にあわせてよいため.

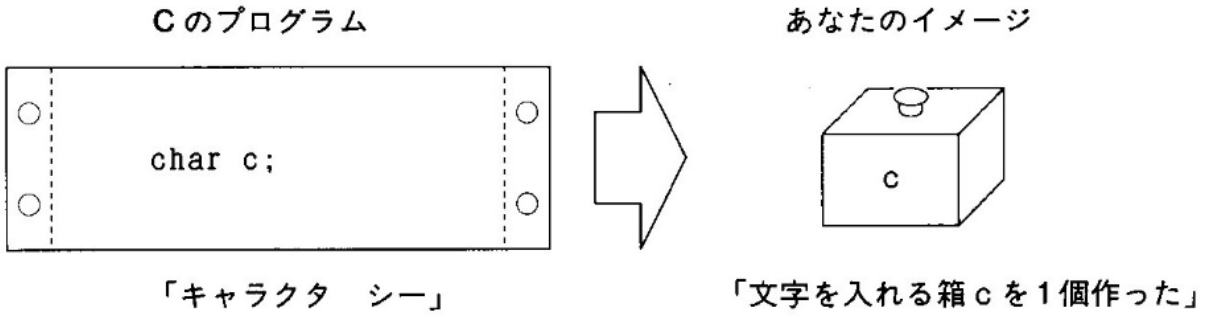




変数を作る (定義もしくは宣言)

- Cでは変数を使う前に必ず変数を定義しなければならない.
- 箱に値を入れる前に箱を準備するのと同じ理屈.
- 前述のように箱の大きさを定義時点で指定する.
- 以下，例



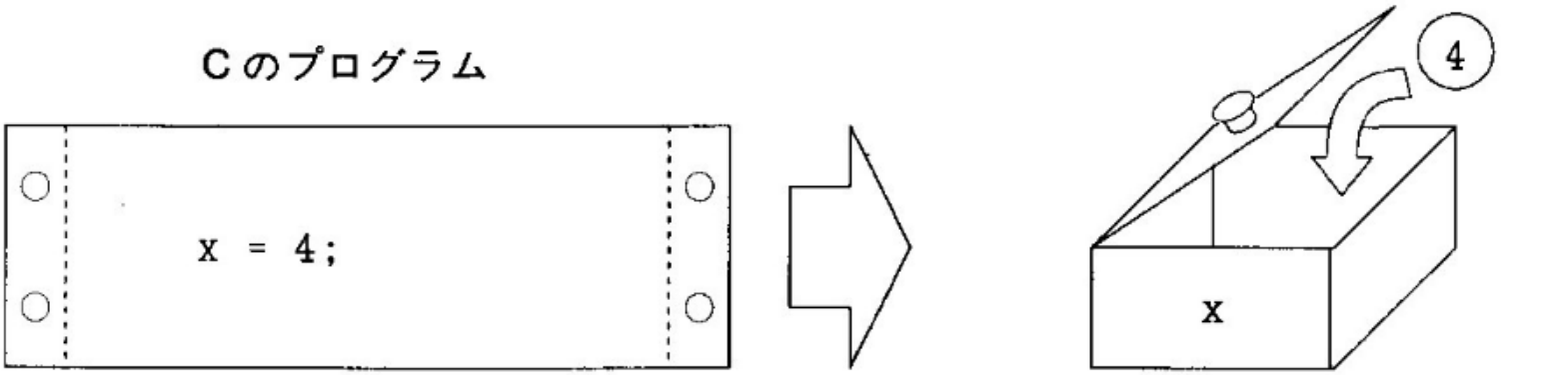


変数に値を入れる; 代入文

- 定義した変数に値を保管するには，以下の構文を用いる，例えば，

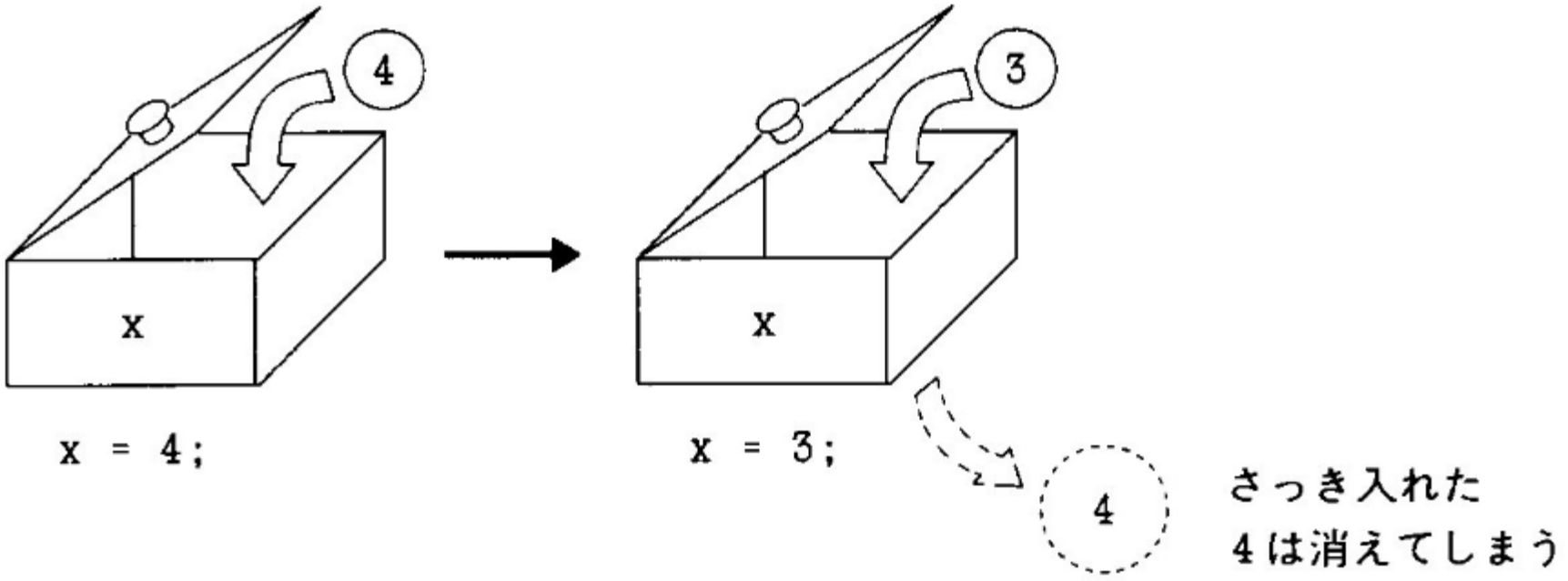
```
x = 4;
```

- 数学の等式と見た目は同じだが意味は違うので注意！
- 必ず = の左には，変数名が1個でなければいけない．繰り返しになるが，代入文は等式では無い．



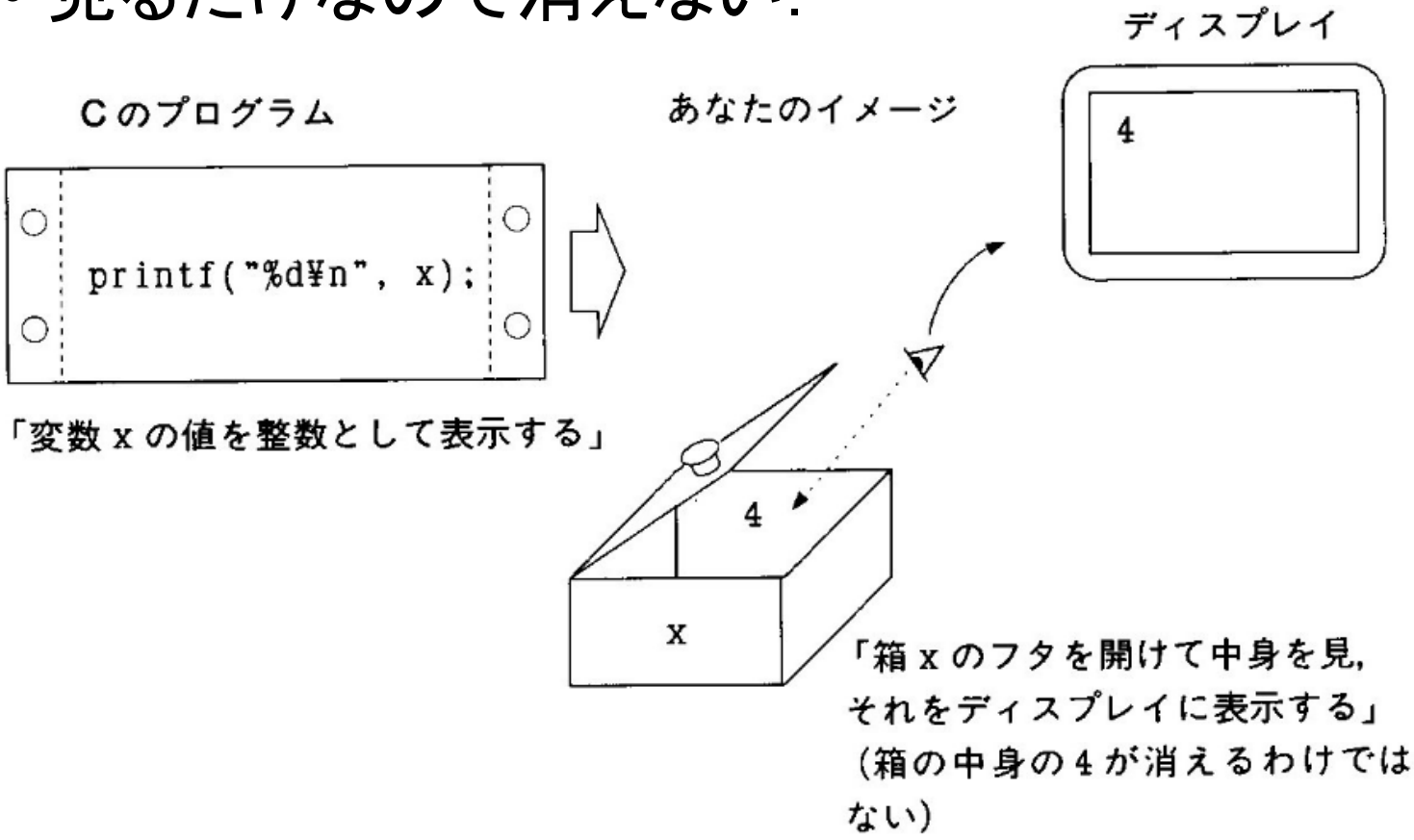
値は上書きすれば前のは消える

- 変数は値の保管場所なので，新しい値を後から入れれば，前に入っていた値は消えてしまう.



変数の値を見る (使う)

- 宣言して値を入れた変数は，通常の値と同様に使うことができる.
- 見るだけなので消えない.



コンピュータの代入文の解釈

- コンピュータは以下の順番で代入文を処理する
  1. = の右側の式を計算する.
  2. 計算結果を = の左側の変数に保管する.
- よって，以下のような構文は正しい (等式と思うと間違っているでしょ)

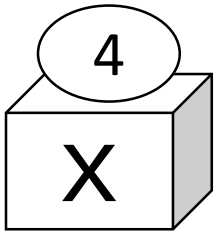
```
x=x+1;
y=y-2;
```

- コンピュータは以下のような書き換えはできない，というか，後者はC言語としては，構文エラー.

```
x=x+1;
x-1=x; // コレは文法的に間違い
```

例題 3-1

```
#include <stdio.h>


int main(void)
{
  int x;

 x = 4;
  printf("%d¥n", 2*x+3);
  return 0;
}
```

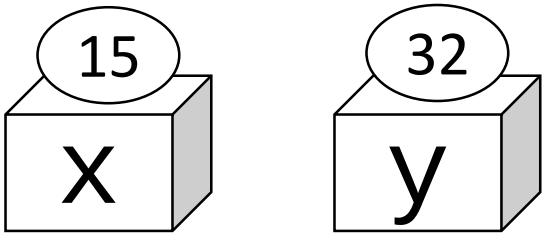


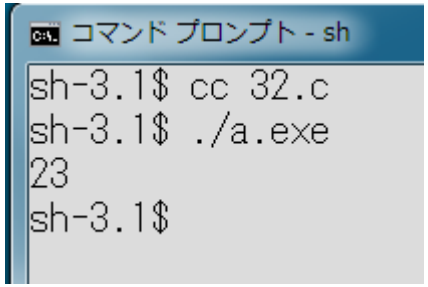
言い忘れたかもしれませんが，
C言語では，
#で始まる行や//を含む行以外は，
空白文字や
改行の位置は自由で結構です．

ただし，日本語の空白文字は，
絶対に使わないでください．
(むしろ日本語はつかわないで)

例題 3-2 複数の変数を利用

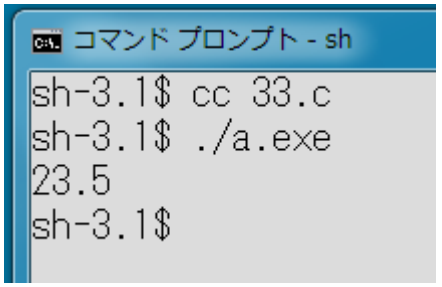
```
#include <stdio.h>

int main(void)
{
   int x, y;
   x = 15;
   y = 32;
   printf("%d¥n", (x+y)/2);
   return 0;
}
```

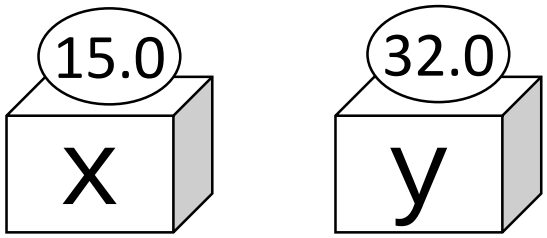




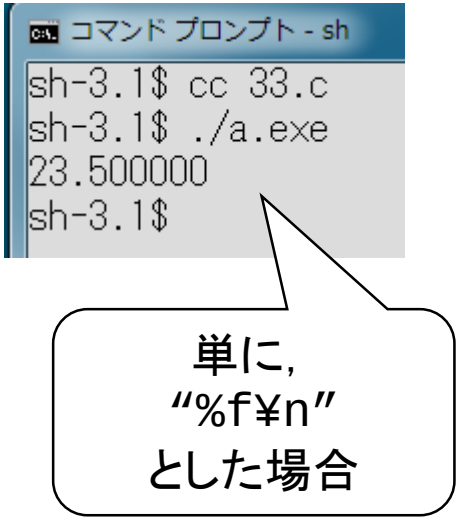
例題 3-3 浮動小数

```
#include <stdio.h>



int main(void)
{
   float x, y;
   x = 15.0;
   y = 32.0;
   printf("%0.1f¥n", (x+y)/2);
   return 0;
}
```



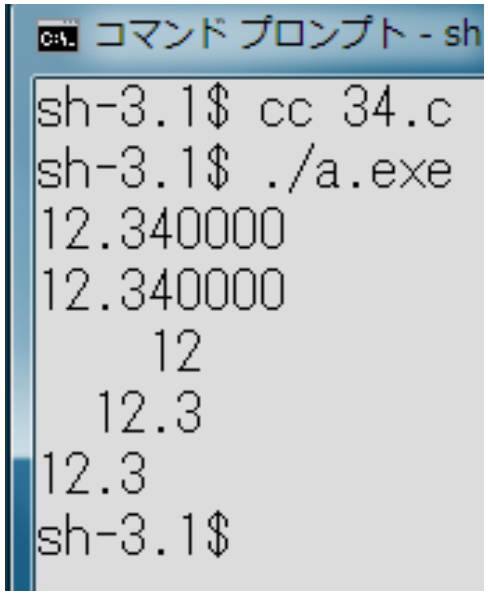


実際，intの箱と
floatの箱の大きさは，
同じ場合が多いです．



例題 3-4 浮動少数の表示法

```
#include <stdio.h>

int main(void);

int main(void) {
   float x=12.34;
   printf("%f¥n", x);
   printf("%6f¥n", x);
   printf("%6.0f¥n", x);
   printf("%6.1f¥n", x);
   printf("%0.1f¥n", x);
   return 0;
}
```



printfのformatについて

- printfのフォーマット指定につかう %d 等は，かなり色々な表現ができます.
- しかし，本授業ではそれを扱いません.
- 単に，%d %f 等で数値等を単純に表示するだけで結構です.

- 今時，実用的なプログラムはGUI (Graphical User Interface)なので，このフォーマットに凝っても仕方が無い.
  - 後述のキーボードからの読み込みについても同様.

キーボードから変数に整数を入力

- キーボードから入力した整数値を変数に保存することができる. (まぁ, できないと困るよね.)
- まず, 保存するための変数を定義する.
- 以下に示すような, scanf という関数を用いると, キーボード入力を指定した変数に入れられる.
- 変数の前の & は忘れないように. 詳細は後日に.
- 教科書(レ)には違うやり方が書いてありますが, 無視してください.

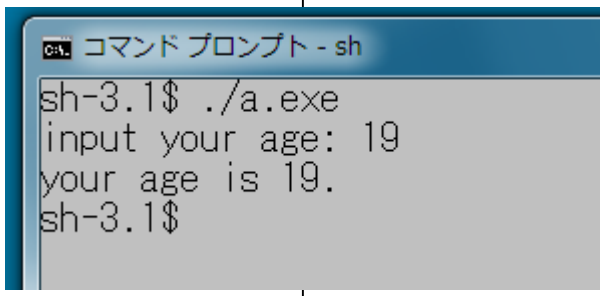
```
#include <stdio.h>

int main(void){
  int val;
    scanf("%d", &val);
return 0;
}
```

p.14-15
(明)

(レ)では
複雑な方法を
書いてるが
お勧めしない

整数値入力の簡単な例

```
/*
 age.c
 input your age
 */
#include <stdio.h>

int main(void){
    int age;

    printf("input your age: ");
    scanf("%d", &age);

    printf("your age is %d. ¥n", age);
    return 0;
}
```



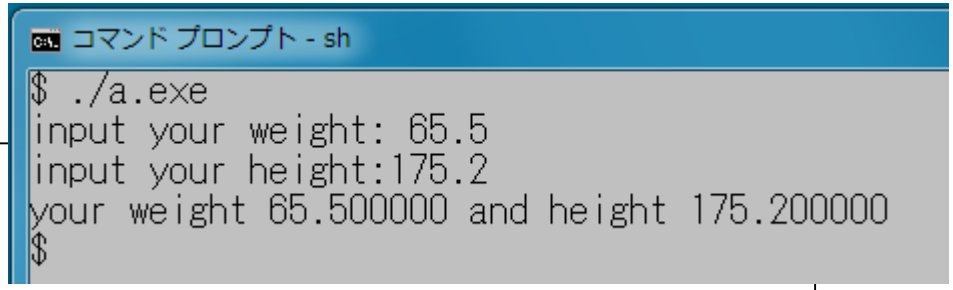
実数値入力の簡単な例

```
#include <stdio.h>

int main(void){
    float h, w;

    printf(“input your weight: “);
    scanf(“%f", &w);
    printf("input your height:");
    scanf("%f", &h);

    printf("your weight %f and height %f¥n", w, h);
    return 0;
}
```

```
コマンド プロンプト - sh
sh-3.1$ ./a.exe
input your weight: 65.5
input your height:175.2
your weight 65.500000 and height 175.199997
sh-3.1$
```

Float は精度が悪い例

floatでなくdoubleなら平気




```
#include <stdio.h>

int main(void){
    double h, w;

    printf("input your weight: ");
    scanf("%lf", &w);
    printf("input your height:");
    scanf("%lf", &h);

    printf("your weight %lf and height %lf¥n", w, h);
    return 0;
}
```

プログラム中の注釈 コメント文

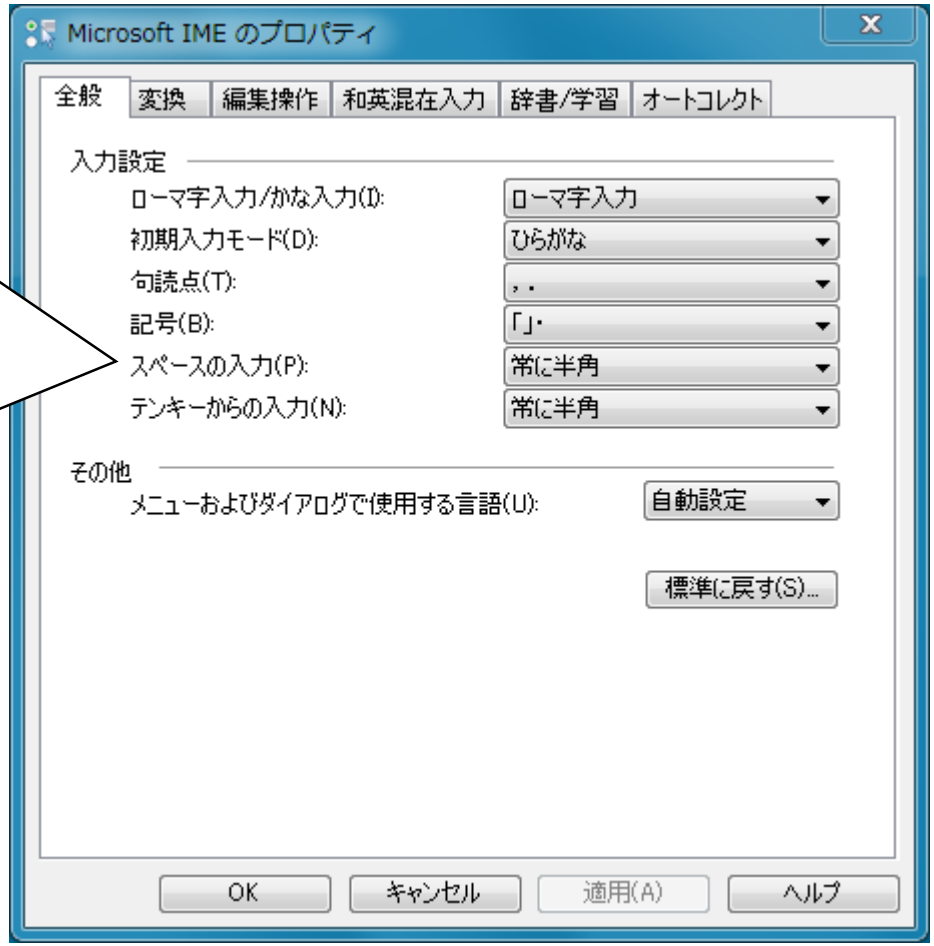
- プログラム中に，人間が後から内容を確認するためのメモを書くことができる.
- これをコメント文と呼ぶ.
- コメント文はメモなのでコンパイル等のコンピュータの動作へは影響を及ぼさない.

- C言語(C99以降)では二種類の書き方が可能
  - /* と */ の間に書く. 複数行にわたり記述可能.
  - // から行末まで書く. 一行のみ.
- 本授業ではコメント文中だけには日本語を書いてもよい. (日本語の空白文字は使わないように)

参考 全角スペースは無効化がベター

少なくとも日本語の空白文字はプログラムでは害悪しかないので，無効化するのがよいと思います．

まぁ，氏名の区切りとかで全角スペースを強要する残念なサイトとかもありますが．orz



コメントの例

```
/*
平均値を計算
Author: kaiya
Date: 18/10/2014
*/
#include <stdio.h>

int main(void){
  float x, y; // floatは精度が悪いのでお勧めしません
  x = 15.0;
  y = 32.0;
  printf(“%0.1f¥n”, (x+y)/2);
  return 0;
}
```

プログラムの空白や改行について

- # で始まる行を除き，プログラムは改行や空白は自由に入れてかまいません．
- // でコメントがある行も注意してください．
- 以下は同じプログラムですが，読みにくい右はお勧めしません．

```
#include <stdio.h>

int main(void)
{
  int x, y;
  x = 15;
  y = 32;
  printf("%d¥n", (x+y)/2);
  return 0;
}
```

```
#include <stdio.h>
int main(void){int x,y;x=15;y=32;
printf("%d¥n",(x+y)/2);return 0;}
```

演習問題2

- 半径を整数値でキーボードから入力すると，おおよその円の面積と円周の長さを表示するプログラムを作成せよ.
- 円周率は「3」を利用せよ. (「おおよそ」ゆえ)
- ソースプログラム名は circle.c としてください.
- dotcampusにアップしてね.

期待される結果の例

```
$ cc circle.c
$ ./a
5
area = 75 circumference = 30
$
```

コレはユーザーが5と入力して，エンターを押すものとしてください.

本日は以上