精密工学科プログラミング基礎Ⅰ

第5回資料

今回の授業で習得してほしいこと:

- 関数
  - 定義・呼出の方法
  - 変数のスコープ
- ポインタ
  - 意味
  - 関数へのポインタ渡しの方法

資料のURL : http://www.den.t.u-tokyo.ac.jp/prog/

関数とは？

- ある計算式(処理)を
  入力を変えて複数回行いたいときに便利

数式

f:(実数,実数) → 実数
$f(x,y) = x^2 + y^2$

C言語

```
double f(double x, double y) {
  double z;
  z = x*x + y*y;
  return z;
}
```

戻値(出力)の変数型

値を戻す命令

引数(入力)の変数型の宣言

- sin 関数 “double sin(double x)” などは <math.h> の中に定義がされている
- “int main(void)” も関数であり、C言語では最初に呼び出される関数
  - “void” は引数なしを表す
  - 最後に “return” する値は、0なら正常終了、-1なら異常終了を表す。

関数の定義のしかた

- main 関数の外側に定義する
  - ケース㊤: main より上に書くなら定義だけ
  - ケース㊦: main より下ならプロトタイプ宣言だけ上に書く

ケース㊤

```
#include <stdio.h>

double f(double x, double y){
  double z;
  z = x*x + y*y;
  return z;
}

int main(void){
   ・・・
}
```

ケース㊦

```
#include <stdi

double f(double x, double y);

int main(void){
   ・・・
}

double f(double x, double y){
  double z;
  z = x*x + y*y;
  return z;
}
```

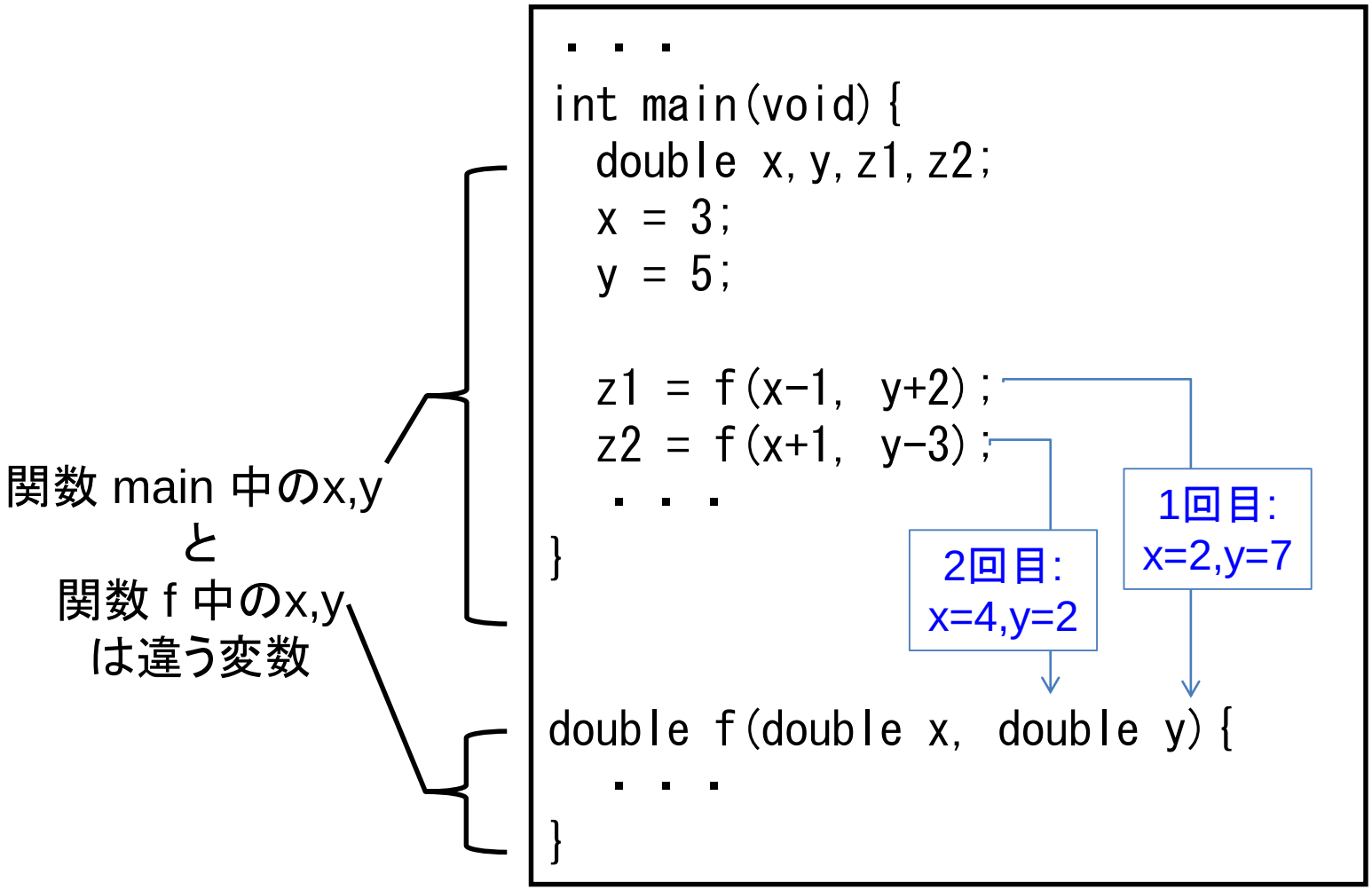
C言語コンパイラは
未宣言の関数・変数の
解釈を後回しにできない

プロトタイプ
宣言

こちらの方が一般的 →

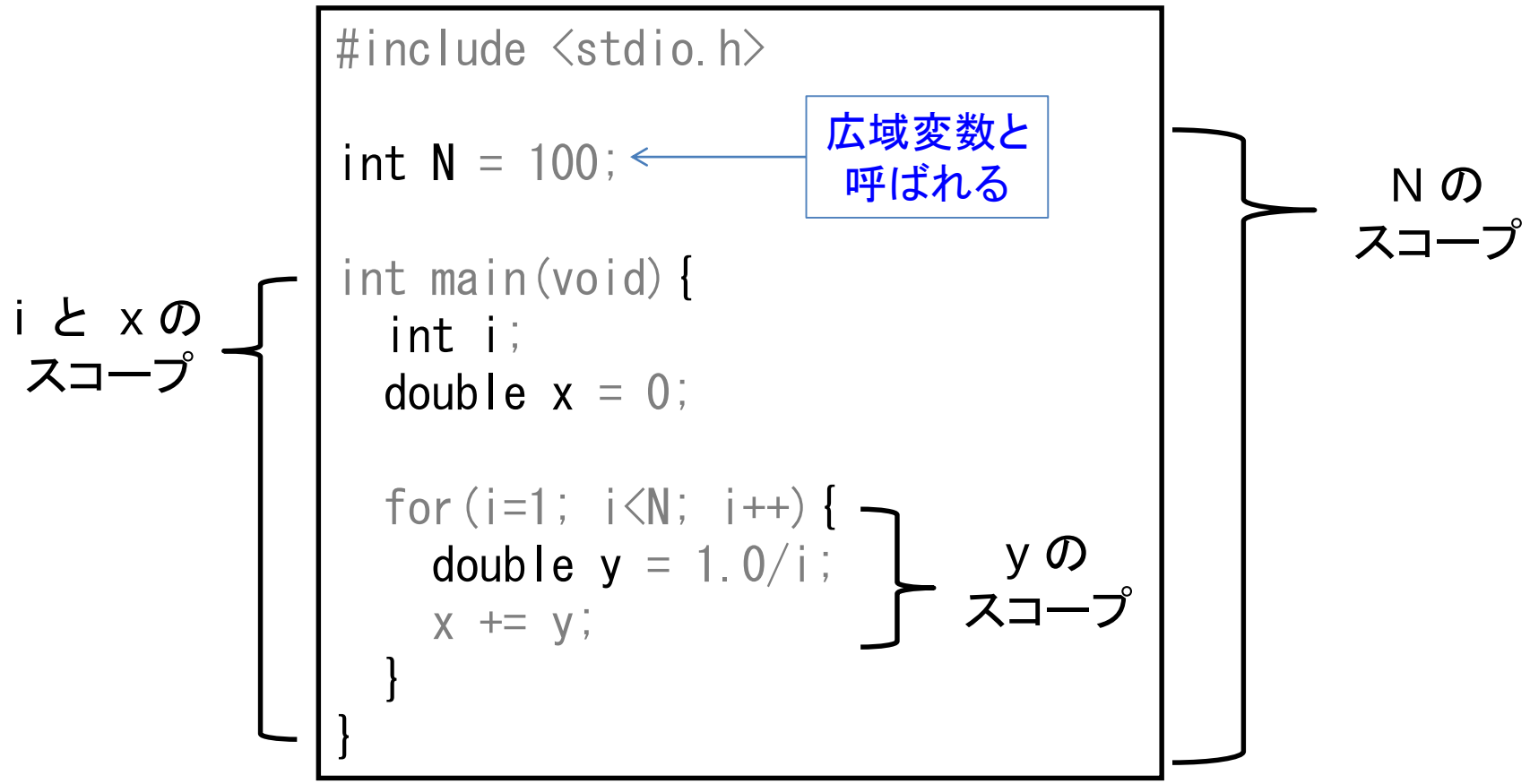
関数の呼び出し

- いままで使ってきた関数と同じ
- 同じ変数名でも、関数が違えば別物



変数のスコープ

- 中括弧 “{ … }” の中で宣言したら、
  その中でのみ利用することができる



2つの引数を交換する関数を作りたい

- 以下の関数は機能しません

```
・・・

int main(void){
  int a = 1;
  int b = 3;
  swap(a,b);
  printf( “%d, %d¥n” , a, b);
}

void swap(int x, int y){
  int t = x;
  x = y;
  y = t;
}
```

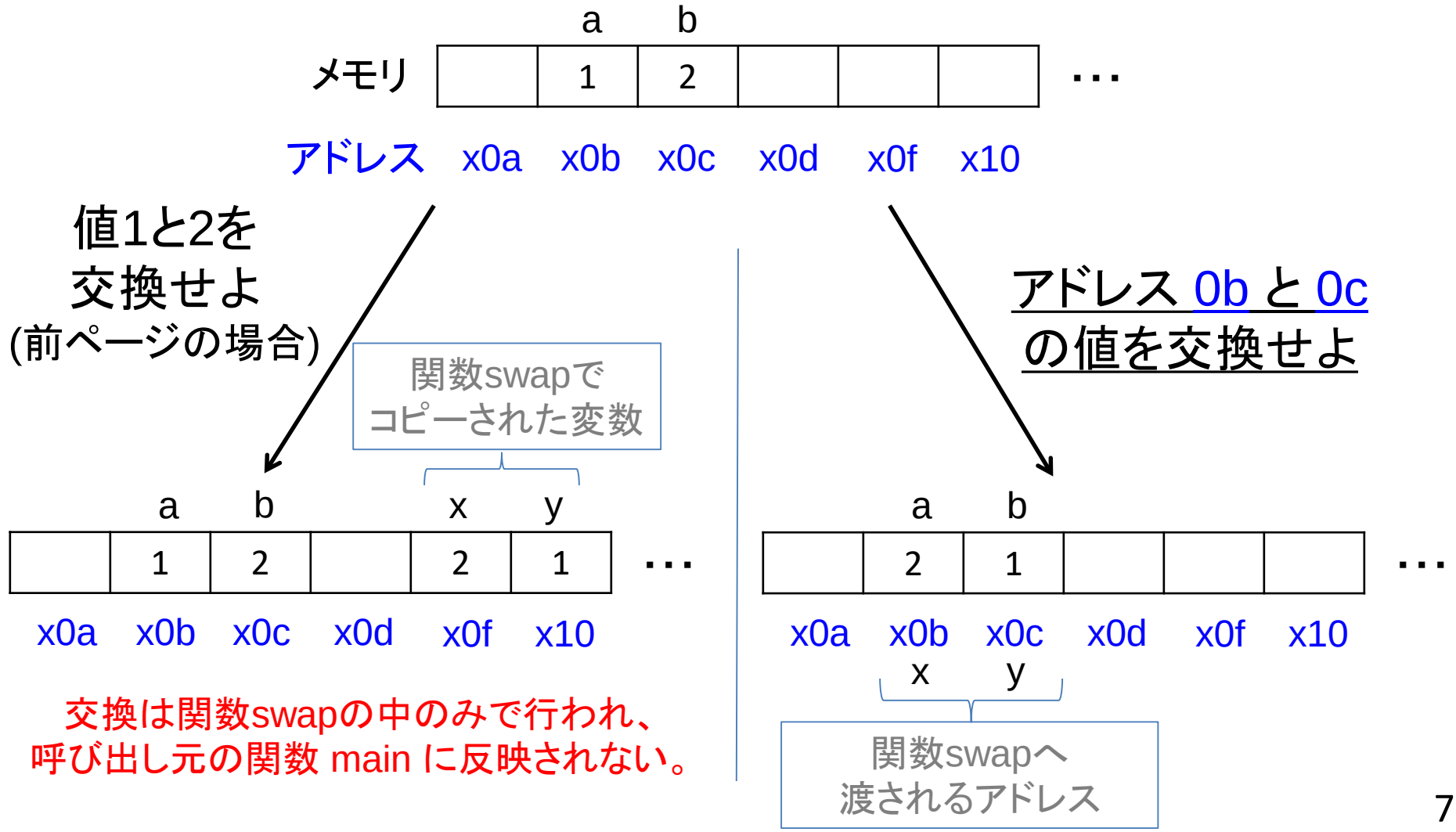
1, 3
と表示され、
交換されない

xとyを交換する
関数
(戻値はなし)

理由：関数 swap の変数は呼び出し時にコピーされるから
（広域変数を使えばできるが、入力を変更して何度も呼び出すことはできない)

では、どうすればよいか？

変数の場所（アドレス）を使う



アドレスの扱い方

- 普通の変数 (int 型の場合)
  - 宣言 int a
  - 値 a
  - アドレス &a (すでにscanf で利用済み)
    - scanf(“%d”, &a); //関数で a に数値を代入

- アドレスへのポインタ変数 (int 型の場合)
  - 宣言 int* p
  - 値 *p
  - アドレス p

同じ * 記号なのに意味が違うので注意
(これがポインタに関する混乱の元凶 !)

アドレス渡しを用いた 2つの値を交換する関数

- ポインタ変数とわかるように “p_” などを付けるとよい

```
・・・

int main(void){
  int a = 1;
  int b = 3;
  swap(&a,&b);
  printf( “%d, %d¥n” , a, b);
}

void swap(int* p_x, int* p_y){
  int t = *p_x;
  *p_x = *p_y;
  *p_y = t;
}
```

int 型のアドレスを渡す

入力はアドレスなので int* 型で宣言

ポインタの指す値なので＊を付ける