情報基礎BII

柳本・野津
TA 中島

関数はなぜ必要か？

- $f(x)=x^2$、$g(x)=x$とするとき、0,1, 2, 3, 4, 5, 6, 7, 8, 9における$f(x)-2g(x)$、$2f(x)+g(x)$、$2f(x)+3g(x)$の値を求めなさい

List1

```
#include <stdio.h>

int main(void){
  int x, a, b, c;

  for(x = 0; x < 10; x++){
    a = x * x - 2 * x;
    b = 2 * x * x + x;
    c = 2 * x * x + 3 * x;
    printf("%d %d  %d¥n", a, b, c);
  }
}
```

List2

```
#include <stdio.h>

int main(void){
  int x, f, g, a, b, c;

  for(x = 0; x < 10; x++){
    f = x * x;
    g = x;
    a = f - 2 * g;
    b = 2 * f + g;
    c = 2 * f + 3 * g;
    printf("%d %d %d¥n", a, b, c);
  }
}
```

関数はなぜ必要か？

- f(x)やg(x)を変えるとき、List1では3箇所変えなくてはいけないが、List2では2箇所でよい

大規模なプログラムでは変え忘れが発生しやすい

-どこでf(x)やg(x)が使われているかすぐに分からない

-似た式が出てくると区別がつかない

同じ動作をする処理をまとめた方が、大規模なプログラムでミスが減り、可読性が向上

➡ C言語では関数として処理をまとめる

関数

- 関数はある値（引数）を受け取って、処理を行って値（戻り値）を返す処理をまとめたもの

（例）fgets(line, sizeof(line), stdin)

C言語では自分で関数を作ることが可能

ローカル変数

- 関数で同じ変数名を使うとどうなるのか？

  関数の中と外では別のものとして扱われる

  ⇨ローカル変数

- 関数の引数はどうなるの？

  引数も関数の中でのみ有効なものとして扱われる

  引数の値を変えてももとの変数に影響は及ばない

この2つの約束のおかげで、たくさんの人が勝手に関数を作っても問題が発生しない

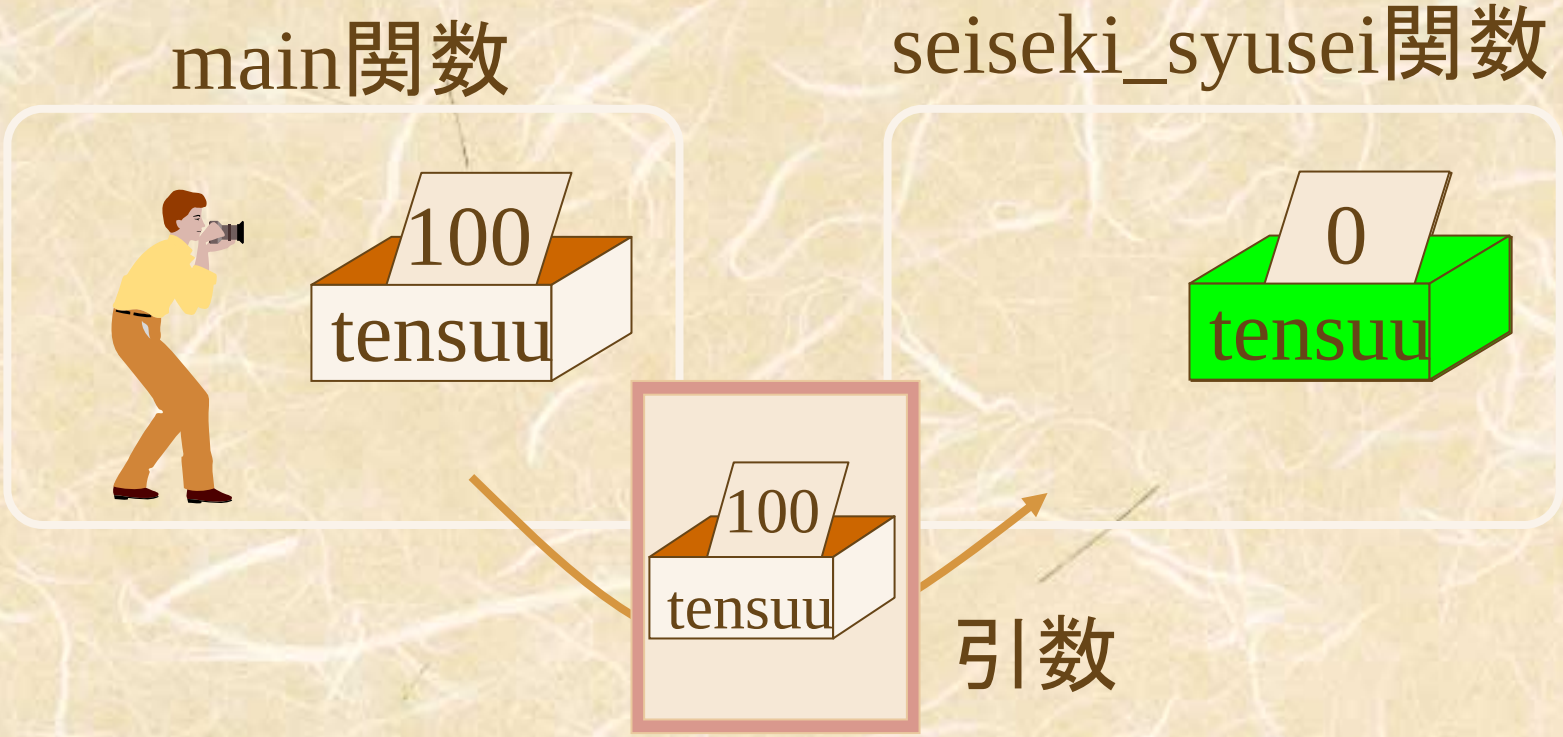
サンプルプログラム

```
#include <stdio.h>

int seiseki_syuusei(int tensuu){
        tensuu = 0;
        return(tensuu);
}

int main(void){
        int tensuu;
        tensuu = 100;
        printf("点数:%d¥n", tensuu);
        seiseki_syuusei(tensuu);
        printf("点数:%d¥n", tensuu);
        tensuu = seiseki_syuusei(tensuu);
        printf("点数:%d¥n", tensuu);
        return(0);
}
```

引数について

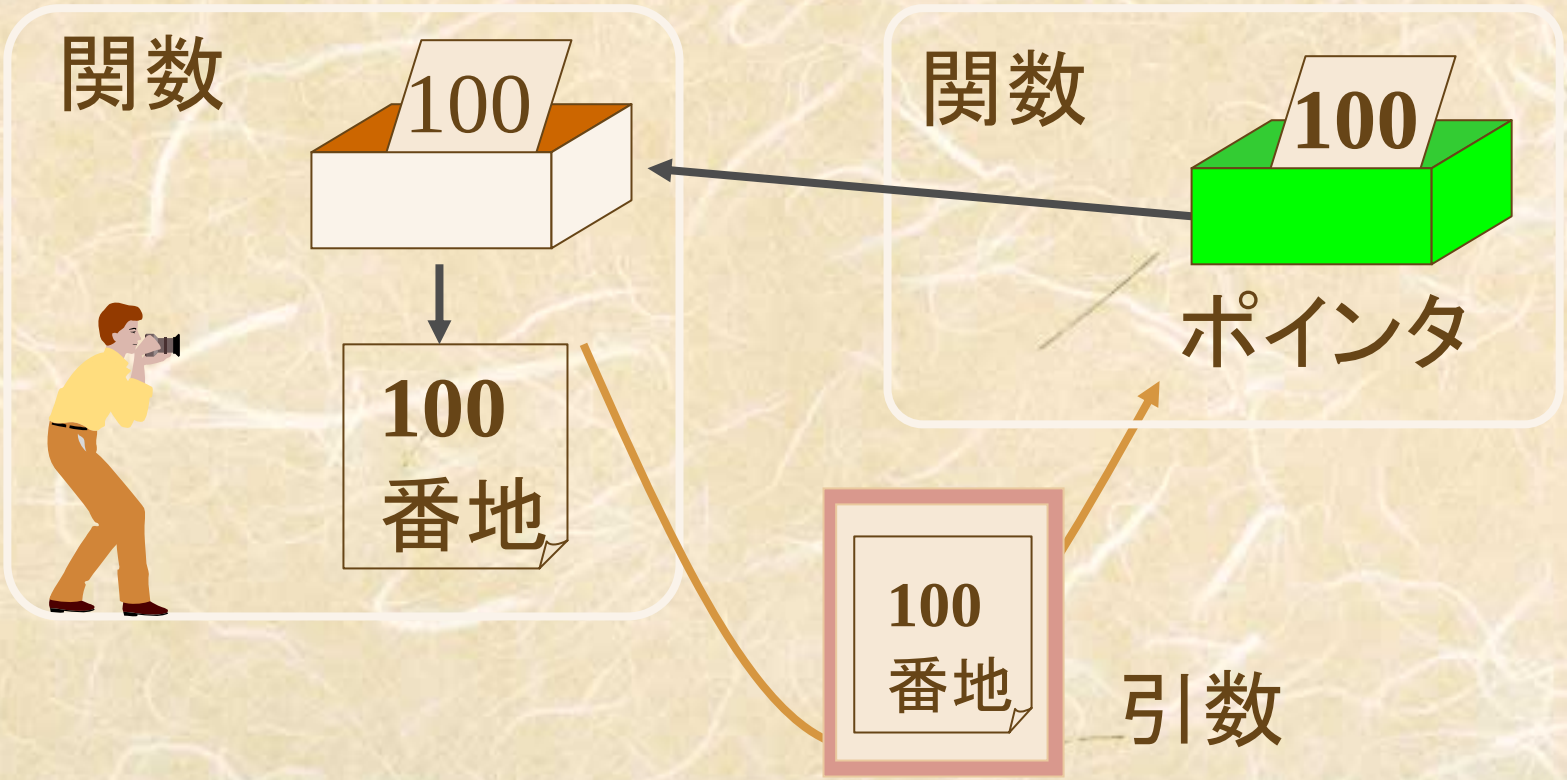


引数はコピーして値を渡すので、関数内で値を変えても
呼び出し先で影響を受けない

ポインタにより引数の受け渡し

- 呼び出し先の変数を変更するときは？

  ⇒アドレスを引数として渡す



配列の渡し方

- 引数に配列を使いたいときは？
  - 引数は変数しか渡せない
    - ⇒ 配列のアドレスを渡すことで問題を回避

関数の定義

```
int hoge(int a[]){
        ⋮
}
```

or

```
int hoge(int *a){
        ⋮
}
```

配列はポインタとしても表現できるので、関数の定義ではどちらを使っても良い。
関数の引数のみ、配列の大きさを指定しない形式（int a[]）が利用できる。
ポインタで配列を渡すので誤って書き換えないように注意すること

サンプルプログラム

```
#include <stdio.h>

#define MAX_BUF 256

int count(char *a){
  int i;

  i = 0;
  while(a[i] != '¥0'){
    i++;
 }

  return(i-1);
}
```

```
int main(void){
  char str[MAX_BUF];
  int nchara;

  printf("Input string:");
  fgets(str, MAX_BUF, stdin);
  nchara = count(str);
  printf("%d characters¥n", nchara);

  return(0);
}
```

二次元配列

- 配列と同じようにアドレスを渡す

関数の定義

二番目の添え字番号

```
int hoge(int *array[3]){
}

int main(void){
  int a[3][3];

  hoge(a);

}
```

二次元配列

```
int hoge(int array[][3]){
}

int main(void){
  int a[3][3];

  hoge(a);

}
```

二次元配列

ポインタのまとめ

ポインタを用いると変数の値が直接操作できるので、return()で実行結果を返す必要がない

⇨ 実際の関数では、関数内での処理が正常に終了したかどうかを確認するため、値を返すのが普通である

ポインタを引数にする必要がある場合

-値の入れ替えなど直接変数の値を操作したい場合
-戻り値を複数返したい場合
-配列を引数として渡したい場合