知能情報工学演習I
第12回（C言語第6回）

岩村雅一

masa@cs.osakafu-u.ac.jp

C言語の予定

*：岩村不在

7. 5月27日* プログラミング環境（テキスト1,2章）
8. 6月 3日* 変数とデータ型（3章）、演算子（4章）
9. 6月10日* コンソール入出力（6章）、配列（3章）、数学処理の標準ライブラリ（11章）
10. 6月17日* 制御文1（テキスト5章）
11. 6月24日 制御文2（テキスト5章）
12. 7月 8日 関数1（テキスト7章）、プリプロセッサ（テキスト10章）
13. 7月15日 応用プログラム

本日のメニュー

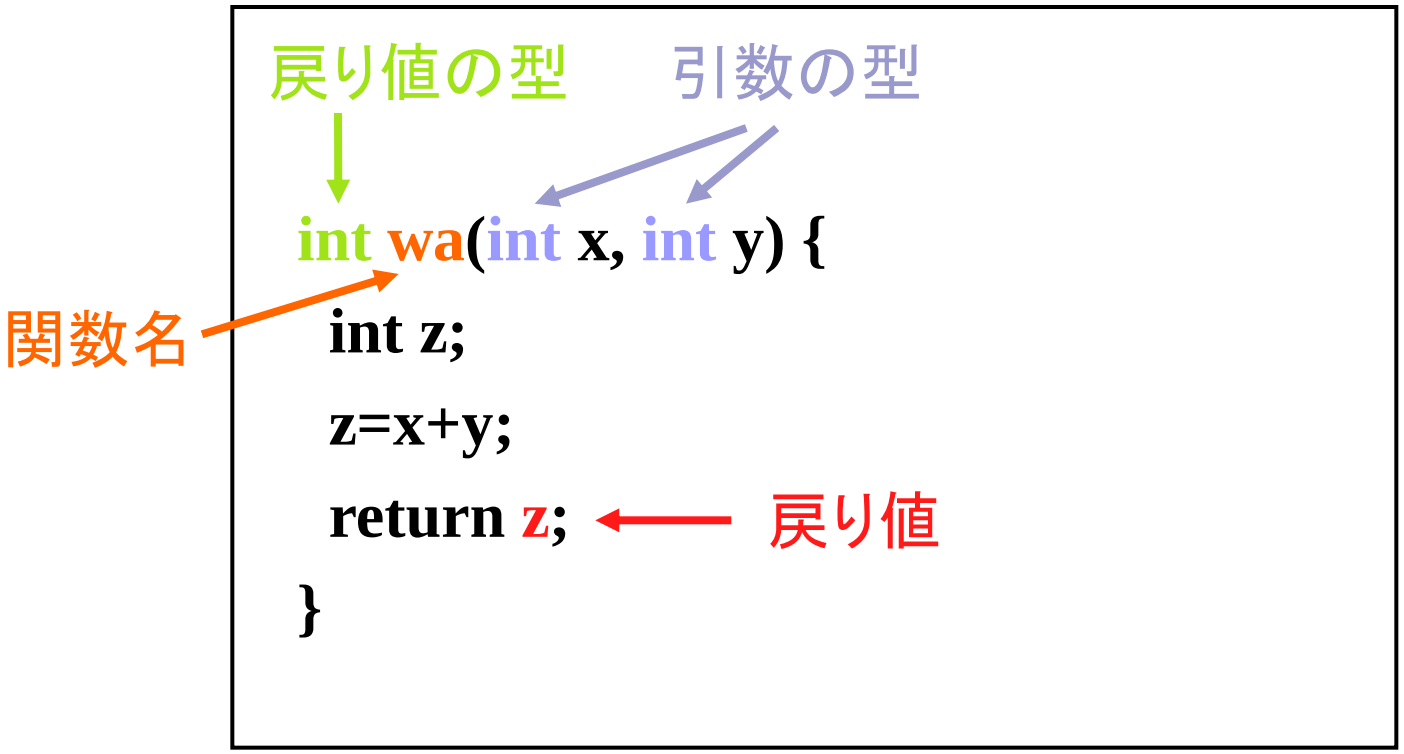
- 関数
  - 関数とは何か
    - 引数や戻り値が無い関数
    - return文
    - 配列を関数に渡す方法
  - ローカル変数とグローバル変数
  - プロトタイプ宣言
  - プリプロセッサ

関数とは

- 引数を取り、処理の後、戻り値を返すもの
  - 数学の関数(”y=f(x)”)のようなもの
  - printf()やscanf()は関数
  - main()も関数

関数の例（定義）

- 関数waは、2つの引数xとyを元にzの値を計算し、戻り値として返す



サンプルプログラム

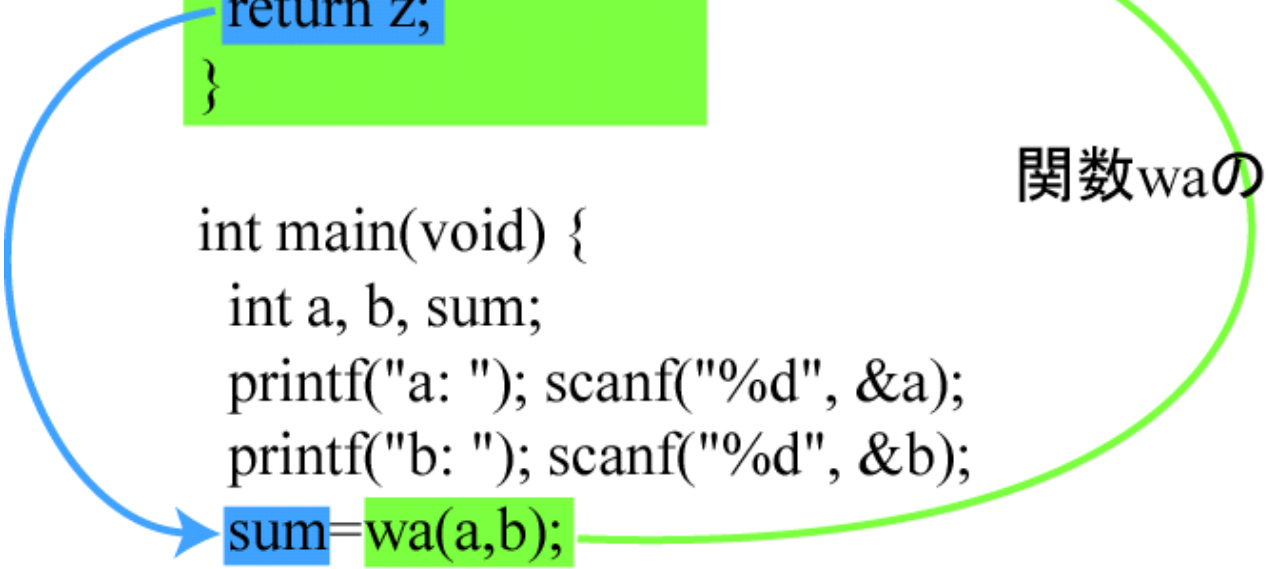
```
#include <stdio.h>

int wa(int x, int y) {
  int z;
  z=x+y;
  return z;
}

int main(void) {
  int a, b, sum;
  printf("a: "); scanf("%d", &a);
  printf("b: "); scanf("%d", &b);
  sum=wa(a,b);
  printf("a + b = %d\n", sum);

  return 0;
}
```

サンプルプログラム

```
#include <stdio.h>

int wa(int x, int y) {
  int z;
  z=x+y;
  return z;
}

int main(void) {
  int a, b, sum;
  printf("a: "); scanf("%d", &a);
  printf("b: "); scanf("%d", &b);
  sum=wa(a,b);
  printf("a + b = %d\n", sum);

  return 0;
}
```

```
int x=a;
int y=b;
```

関数waの呼び出し



引数や戻り値が無い関数

- 引数が無い関数

```
int func(void) {
  …
}
```

- 戻り値が無い関数

```
void func(int a) {
  …
}
```

return文の役割

- return文があると、関数の処理を打ち切って呼び出し元に戻る

```
int waru(int x, int y) {
  int z;
  if (y==0) {
  return -999;
}
  z=x/y;
  return z;
}
```

yが0ならば-999を返して関数の計算は終了

配列を関数に渡す方法

- サンプルプログラムの前に、配列の初期化について

配列の初期化（テキストP.67）

- 変数は宣言するときに値を設定できる

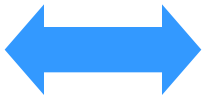
```
int a=10;
```

- 配列は宣言するときに値を設定できる

```
int a[3]={10,20,30};
```

```
int a[]={10,20,30};
```

省略可能



等価

```
int a[3];
a[0] = 10;
a[1] = 20;
a[2] = 30;
```

配列を関数に渡す方法のサンプルプログラム

配列の初期化

配列の大きさは（最後の1つだけ）省略可能

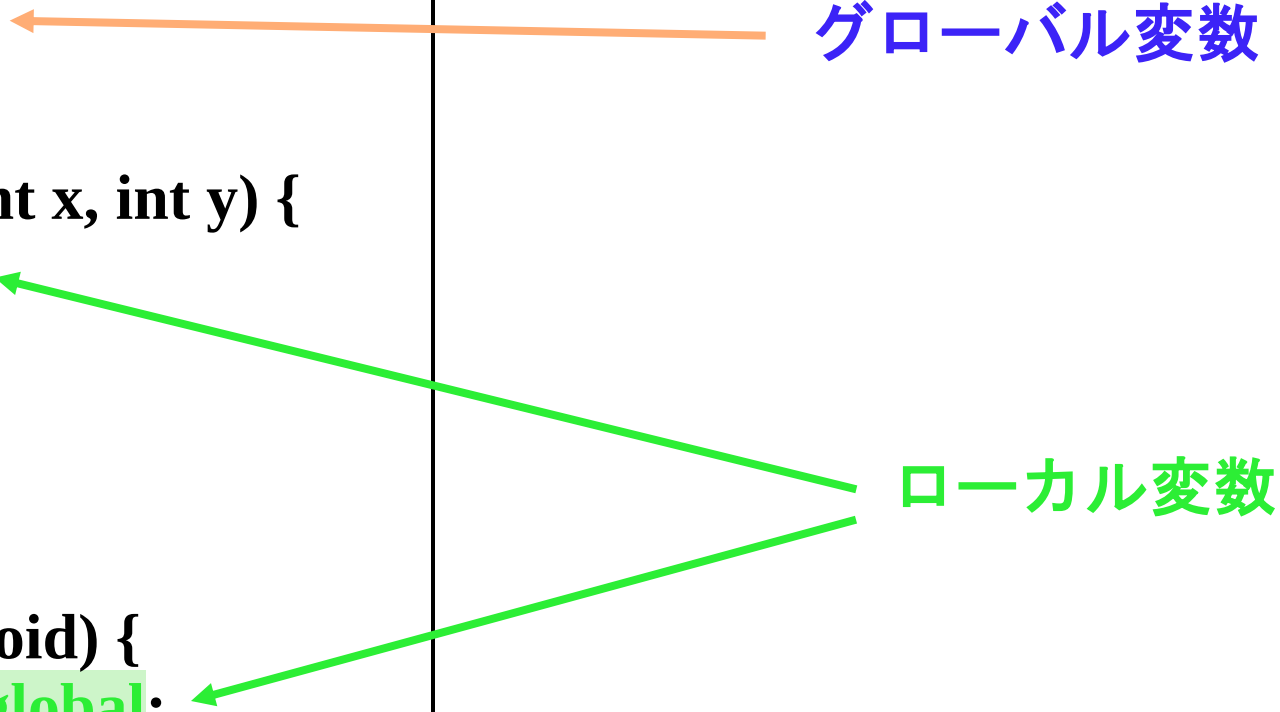
```
#include<stdio.h>

int wa2(int num, int x[]) {
  int i;
  int sum=0;
  for(i=0; i<num; i++) {
    sum+= x[i];
  }
  return sum;
}
```

配列の和

```
int main(void){
  int a[]={1,3,5,7,9};
  int sum;

  sum = wa2(5,a);

  printf("sum = %d¥n",sum);

  return 0;
}
```

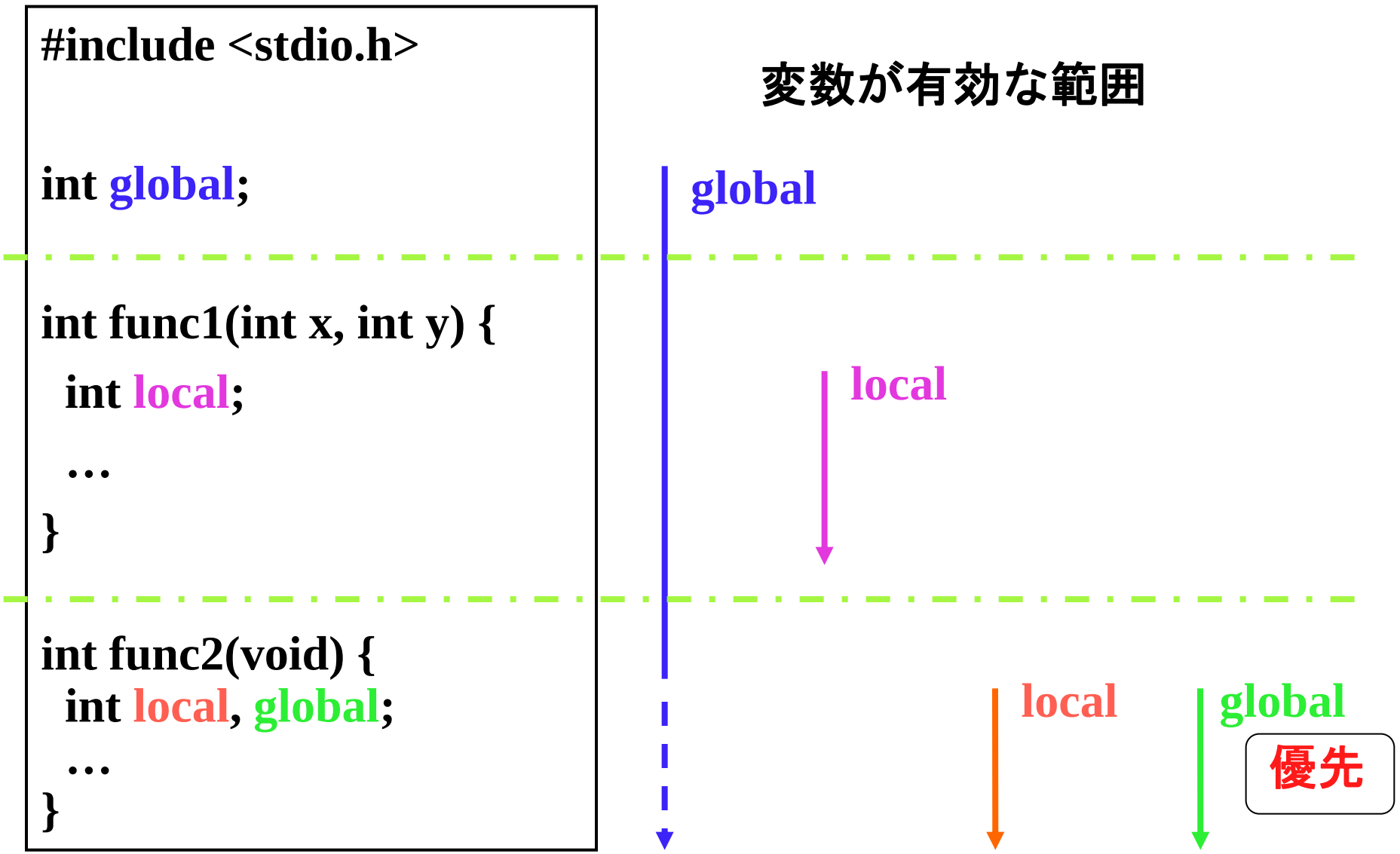
ローカル変数とグローバル変数

```
#include <stdio.h>

int global;

int func1(int x, int y) {
  int local;
  …
}

int func2(void) {
  int local, global;
  …
}
```



ローカル変数とグローバル変数

```
#include <stdio.h>

int global;

int func1(int x, int y) {
  int local;
  …
}

int func2(void) {
  int local, global;
  …
}
```



プロトタイプ宣言

- 関数の「定義」を先にできない場合は「宣言」だけを先に書く

プロトタイプ宣言

移動

```
#include <stdio.h>


int main(void) {
  int a, b, sum;
  printf("a: "); scanf("%d", &a);
  printf("b: "); scanf("%d", &b);
  sum=wa(a,b); ???
  printf("a + b = %d\n", sum);

  return 0;
}

int wa(int x, int y) {
  int z;
  z=x+y;
  return z;
}
```

プロトタイプ宣言

```
#include <stdio.h>

int wa(int x, int y);

int main(void) {
  int a, b, sum;
  printf("a: "); scanf("%d", &a);
  printf("b: "); scanf("%d", &b);
  sum=wa(a,b);
  printf("a + b = %d\n", sum);

  return 0;
}

int wa(int x, int y) {
  int z;
  z=x+y;
  return z;
}
```

プリプロセッサ（テキストP.247）

- コンパイルする前にソースコードを変更する
  - #include …ファイルの挿入
  - #define … 文字の置き換え

#include …ファイルの挿入

※通常はヘッダファイルと呼ばれるプロトタイプ宣言が書かれたファイルを読み込む

ファイル名：何でもいい

```
#include <stdio.h>
#include "wa.c"

int main(void) {
  int a, b, sum;
  printf("a: "); scanf("%d", &a);
  printf("b: "); scanf("%d", &b);
  sum=wa(a,b);
  printf("a + b = %d¥n", sum);

  return 0;
}
```

ファイル名：wa.c

```
int wa(int x, int y) {
  int z;
  z=x+y;
  return z;
}
```

#define … 文字の置き換え

- 記号定数
  - 単なる文字列の置き換え

    # define　M_PI　3.14159265358979323846
- 関数形式マクロ
  - 関数のように働く

    #define　wa(a,b)　a+b

関数形式マクロの例

コンパイル前に
置き換える

sum=a+b;

```
include <stdio.h>
#define wa(a,b) a+b

int main(void) {
  int a, b, sum;
  printf("a: "); scanf("%d", &a);
  printf("b: "); scanf("%d", &b);
  sum=wa(a,b);
  printf("a + b = %d¥n", sum);

  return 0;
}
```

マクロの長所、短所

- 例：#define wa(a,b) a+b
- 長所
  - 変数の型を気にする必要がない
    - int型でもfloat型でも同じように動作する
  - 関数を呼ぶより速い
- 短所
  - 予期せぬ動作をするときがある
  - wa(a,b)/2 は、関数だったら(a+b)/2になったはずなのに、マクロだとa+b/2になる