SIT SHIBAURA INSTITUTE OF TECHNOLOGY

情報処理演習II
No. 7

2023. 10. 17

芝浦工業大学　システム理工学部　機械制御システム学科
担当：桑原

前回のフォロー

- コメントを付けて可読性を上げると、自分の書いた処理の是非が判断しやすい

```
//  100以内の素数をすべて取り出すのプログラム
#include <stdio.h>
int main (){
      int x; int i;
      int number = 1;                    // numberを素数としてマーク
      for ( x = 1; x <= 100; x++ ){  // xの範囲を定義、1 から100までループ：for文A
        for ( i = 2; i < x; i++ ){       // 2から自分(x)までループ：for文B
            if ( x % i == 0 ) {          //  x が i で割り切れるかどうかを判断する
                number = 0;
                break;                   // 素数ではないなら、forB文を抜け出す
            }                             //
        } // for文B
        if ( number == 1 ) {   //素数である
          printf("%d ", x);           //素数を表示
        }
      } // for文A
}
```

関数の必要性（1）

例：エラー処理付き割算を複数回行うプログラムの一部

```
double  a, b, c, x, y, z;
            （中略）
if (b == 0) {
        printf("0 では割れません！¥n");
        c = 0;
} else { c = a / b;          }

if (y == 0) {
        printf("0 では割れません！¥n");
        z = 0;
} else { z = x / y;          }
```

同様な内容が繰返されているが・・・
これは毎回書かないと駄目?

2つの変数（a と b，x と y）の割算を行い、商を変数（c, z）に代入する。
また、割る数（b, y）が 0 のときはエラー処理を行う。

関数の必要性（2）

例：エラー処理付き割算を複数回行うプログラムの一部

```
double  a, b, c, x, y, z;
            （中略）
c = div(a, b);
z = div(x, y);
```

関数を使うとこんなにすっきり！

エラー処理付き割り算を「**関数 div**」にまとめた
（もちろん関数 div の内容は別記する！）

注意：個別に異なる部分は指定している（この場合はa,b,c,x,y,z）

関数とは

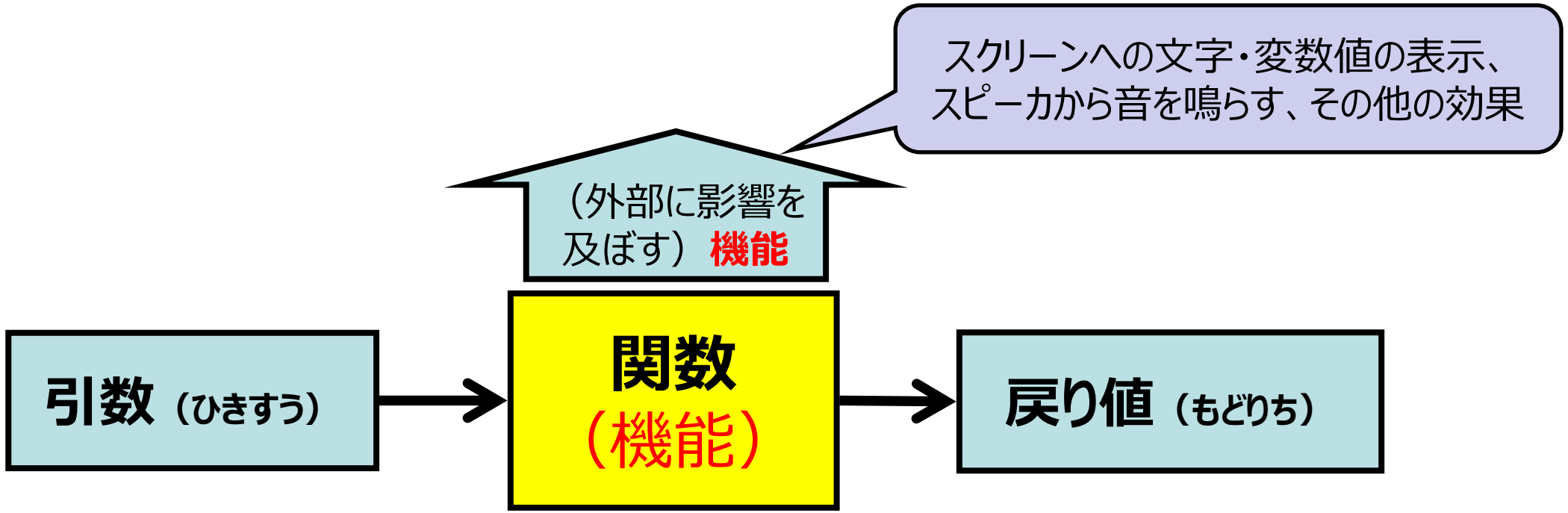
- C言語のプログラム＝関数の集まり
  - printf は画面に文字を表示する関数
  - scanf はキーボード入力を変数に代入する関数
  - sin や cos は正弦関数や余弦関数
  - 実は、プログラムの本体を書いた main も関数の一つ
    C言語のプログラムにはこの main 関数（mainルーチン　とも呼ぶ）が必ず一つだけ存在し、ここからプログラムの　実行が始まる
  - main 関数以外の関数を、単に関数（もしくはsubルーチン）と呼ぶ

- 利用する頻度の高いものを一まとまりにして独立させ，必要に応じて呼出して利用する
  （読みやすさも向上）

関数とは

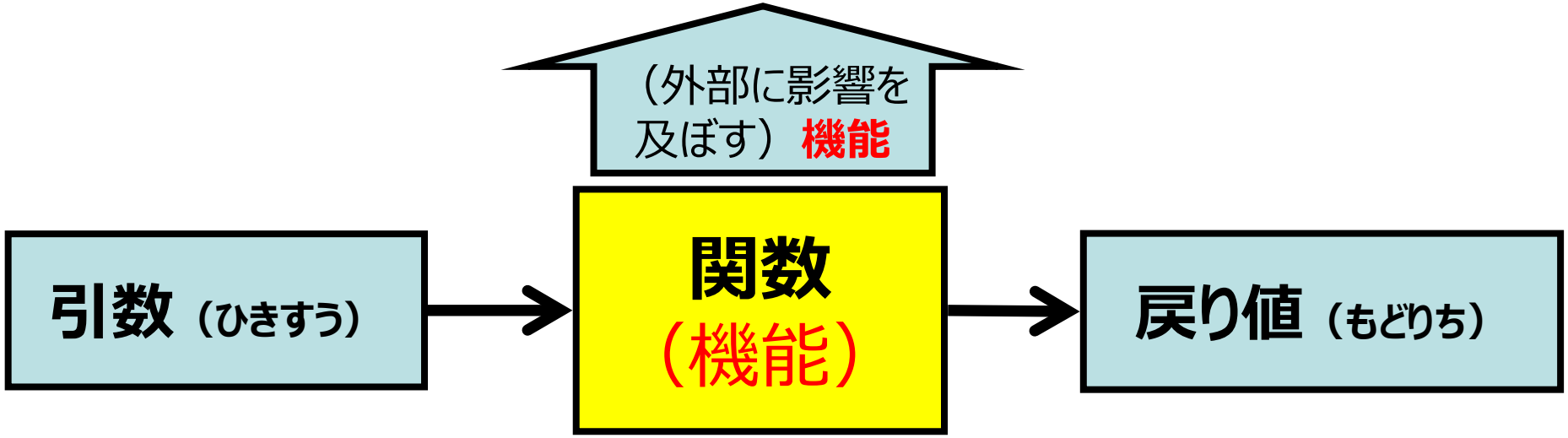
- **関数**は**引数（ひきすう）**を読込み、ある計算や操作を行い、
  戻り値（出力・output とも呼ぶ）を返す

- 関数は戻り値の他にも、**直接外部に影響を及ぼすような機能（効果）**を持つことができる
  （数学で言う「関数」との違い）



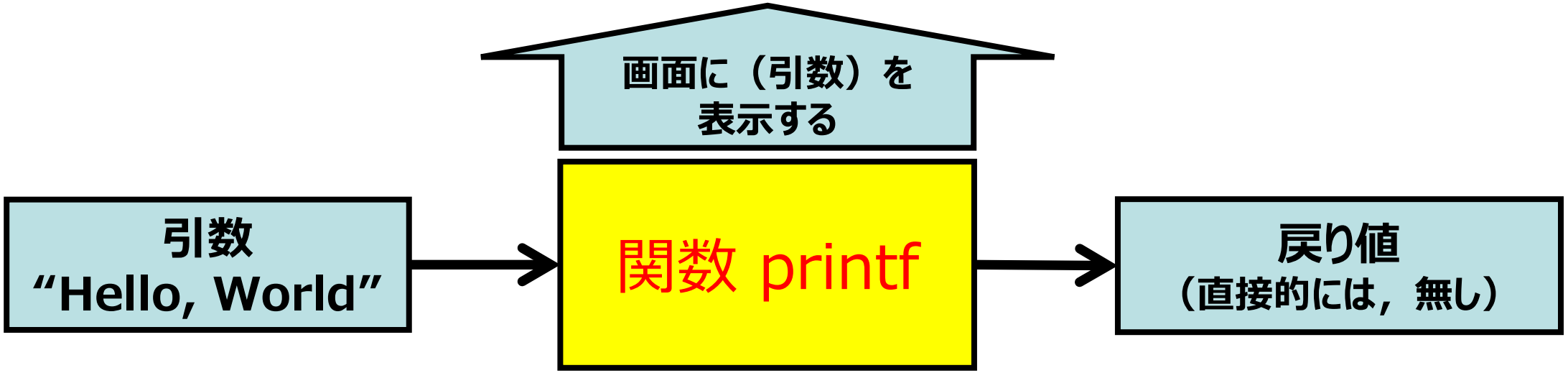
関数とは

- 引数や戻り値は、変数と同様にデータ型を持つ（整数、小数、文字、文字列、・・・）
- データで表せない機能は、「直接外部に影響を及ぼす機能」となる
- 「引数がない関数」や「戻り値がない関数」もある



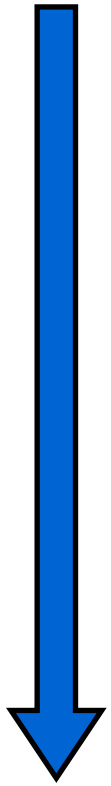
関数とは

- 例1：printf 関数
  - 引数＝「画面に表示したい文字列」
  - 戻り値＝「（直接的には）無し」
  - 直接外部に影響を及ぼす機能＝「画面に（引数）を表示する」



関数を使うための手順

ソース中には
この順で書く



- **関数の定義**
  - 関数の**名前**
  - **引数**宣言（引数のデータ型，引数の名前）
  - 戻り値の**データ型**
  - 関数の機能（実際の処理内容）
- **mainルーチンから関数を呼び出す**
  - 引数を与える
  - 戻り値を回収する

関数の定義

```
戻り値のデータ型　　関数名（引数宣言）
{
        関数の機能（処理内容）;  ←（今までのように複数行書ける）
        return（戻り値）;  ←（main関数との大きな相違点）
}
```

- **関数名は他の変数名と重複**しないようにする
- 引数宣言の例：「int　x」 複数の引数を持つ関数では、**引数宣言を列挙**する。
  例：「int　x, int　y」
- **引数を持たない場合は、引数宣言の場所に「void」と書く**
- **戻り値を持たない関数では、戻り値のデータ型を「void」（void型）とし、「return（戻り値）;」を省略**
- 「直接外部に影響を及ぼす機能」は「関数の機能」として記述

関数の定義の例（1）

```
double   div(double  a, double  b)
{
        double   c;

        if(b == 0) {
                printf("0では割れません!!! ¥n");
                c = 0;
        } else {
                c = a / b;
        }
        return(c);
}
```

関数の定義の例（1）の解説

- 関数の名前は「div」
- 引数は double 型変数 a と double 型変数 b の2つ
- 戻り値のデータ型は double 型
- 関数の機能は ・・・（略）・・・
- **変数 c は関数 div の内部だけで定義された変数.**
  この変数は div の内部（｛から，｝まで）でのみ有効で，
  div の外からは参照も代入もできない（変数の独立性）
- 変数 c の値を戻り値として出力する
- 直接外部に影響を及ぼす機能は
  「画面に・・・（略）・・・を表示する」

関数の定義の例（2）

```
int   max(int  x, int  y)
{
        int  m = x;

        if ( m < y )   m = y;
        return(m);
}
```

- 関数の名前は「max」
- 引数は int 型変数 x と int 型変数 y の2つ
- 戻り値のデータ型は int　（＝関数maxの型）
- 関数の機能は・・・（略）・・・。「return (m);」＝「変数 m の値を戻り値とする」。

  「直接外部に影響を及ぼす機能」は無し

関数の定義を書く場所

main関数の前に関数の定義を書く

```
#include  <stdio.h>

int  max(int  x , int  y)
{
     int  m = x;

     if( m < y )  m = y;
     return(m);
}

main()
{
        ・ ・ ・
}
```

関数 max の定義

main 関数
（プログラムはここから実行が始まる，つまり我々もここからプログラムを読み始めればよい）

関数の呼び出し（1）

```
#include  <stdio.h>

int  max( int x,  int y )
{
        int  m = x;

        if ( m < y )   m = y;
        return ( m );
}

void main()
{
        int  a,  b,  m ;

        a = 1;
        b = 2;

        m =  max(a, b) ;
        printf("%d と %d の最大値は %d です．¥n", a, b, m);
}
```

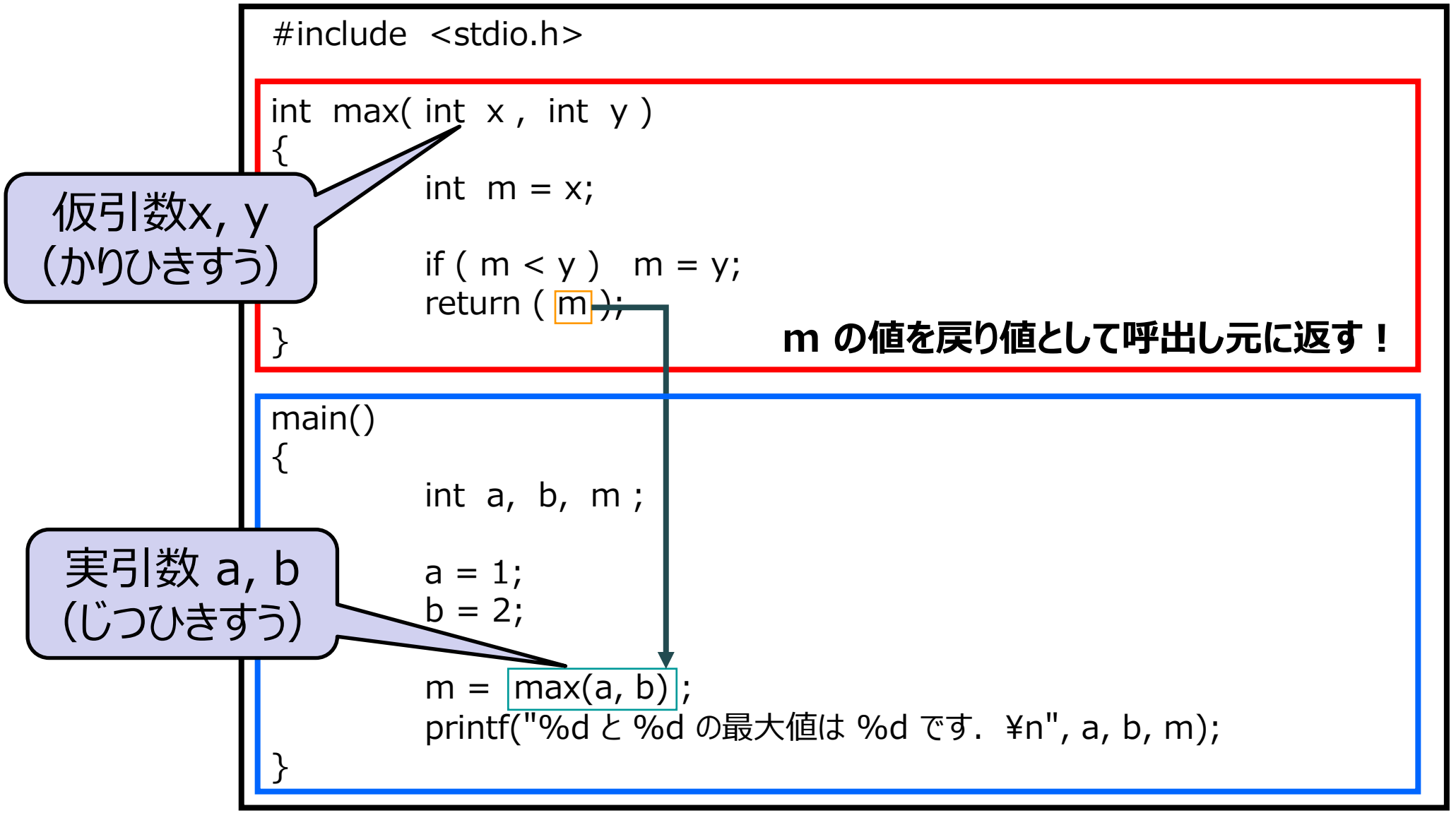
x に 1 を、y に 2 をコピーして呼び出し！

○の変数は関数max 内でのみ有効。
外からは参照も代入もできない
→ main関数内の変数とは関係なし!
（main関数内と同じ変数名も使える）

□の変数は main 内でのみ有効。
外からは参照も代入もできない

関数maxの呼び出し！
引数に変数a**の値**（つまり1）と変数b**の値**（つまり2）を渡している。
よって実際はmax(1,2)を実行する。

関数の呼び出し（2）


#include <stdio.h>
int max(int x , int y)
{
int m = x;
if (m < y) m = y;
return (m);
}
m の値を戻り値として呼出し元に返す！
main()
{
int a, b, m ;
a = 1;
b = 2;
m = max(a, b) ;
printf("%d と %d の最大値は %d です. ¥n", a, b, m);
}
仮引数x, y
（かりひきすう）
実引数 a, b
（じつひきすう）

サンプルプログラム7.1

```
 1    #include  <stdio.h>
 2
 3    void  hello(void)   /* 関数helloの定義(引数，戻り値ともなし) */
 4    {
 5       printf("こんにちは¥n");
 7    }
 8
 9    int  max(int x, int y)    /* 最大値を求める関数maxの定義(引数，戻り値ともあり) */
10    {
11       int max=x;
12
13       if(max<y) max=y;    /* 最大値の入替え */
14         return(max);       /* 呼出し元に結果を戻す(戻り値) */
15    }
16
17    void  print_num(int x, int y)   /* 入力値を表示する関数print_numの定義 */
18    {                                /*                    (引数あり，戻り値なし) */
19       printf("入力された整数は %d と %d ですね．¥n", x, y);
20    }
21
```

```
22    void main()       /* main関数 */
23    {
24       int a,b,c;
25
26       hello();   /* 関数helloの呼出し(引数，戻り値ともなし) */
27
28       printf("一つ目の整数を入力して下さい．¥n");
29       scanf("%d",&a);
30
31       printf("二つ目の整数を入力して下さい．¥n");
32       scanf("%d",&b);
33
34       print_num(a,b);   /* 関数print_numの呼出し(引数あり，戻り値なし) */
35
36       c=max(a,b);       /* 関数maxを呼出し，結果を変数cに格納(引数，戻り値ともあり) */
37       printf("%d と %d の最大値は %d です．¥n", a, b, c);
38    }
```