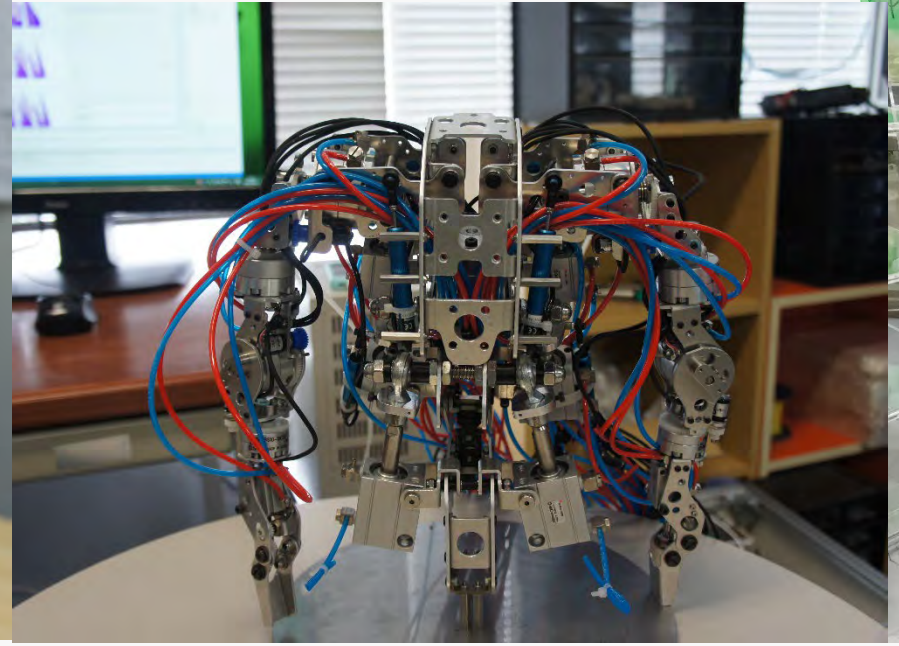




計算機とプログラミング

第３回　基本文法２

担当：石原尚
機械工学専攻　動的システム制御学領域　講師

非常に便利な「関数」について学びます

複雑なソースコードをシンプルに書いていけるようになります

第１回
基礎を学ぶ準備
- 授業の進め方
- C言語の立ち位置
- C言語の環境整備

杉本先生

第２～４回
使い方の基礎
- 変数と制御構造
- ＋関数
- ＋配列

今日はここ！

先週学んだことも踏まえます

石原

第５～８回以降
少し難しい使い方
- ポインタ
- データ構造

杉本先生＆平原先生

第９～１３回
応用と実践
- アルゴリズム
- 総合演習

中西先生＆杉本先生＆石川先生

関数の「役割」「特徴」「使い方」を学びます

そもそも論

学習内容①

関数の役割

- 関数とはどんなものか
- どんな関数があるのか
- なぜ関数が便利なのか

知らないと正しく使えない

学習内容②

関数の特徴

- 関数を実装する方法
- 関数の変身とは
- 関数内は独立世界
- 関数間の値の受け渡し

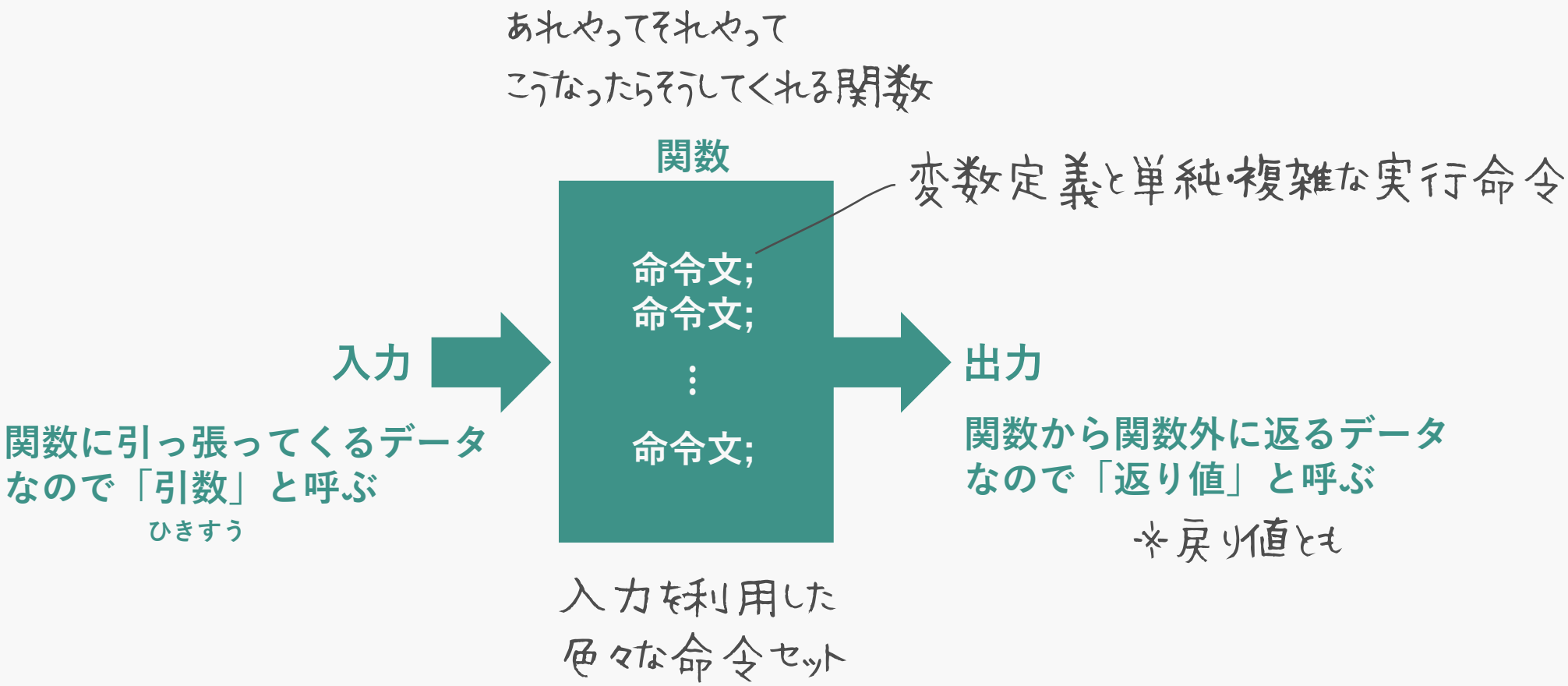
知って上手く使いこなそう

学習内容③

関数の使い方

- 関数の再利用
- 型の予告宣言
- 関数の中の関数
- 返り値の扱い

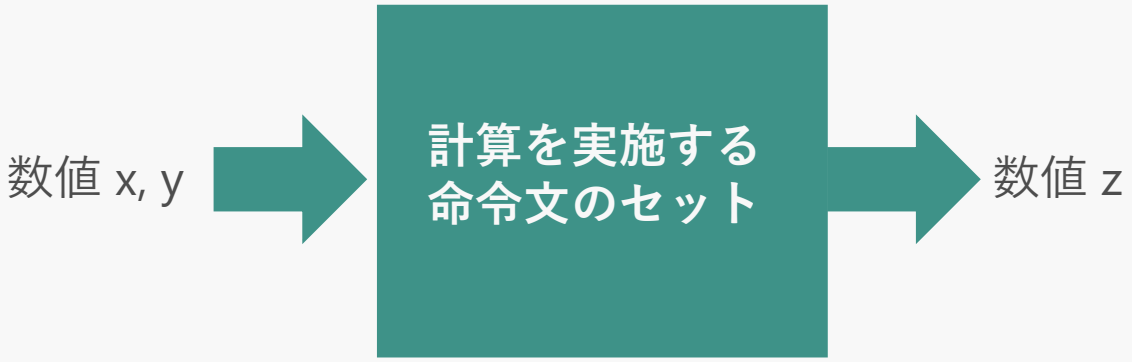
関数とは「入出力機能付きの命令文のセット」



数学的な意味での「関数」よりも幅広い

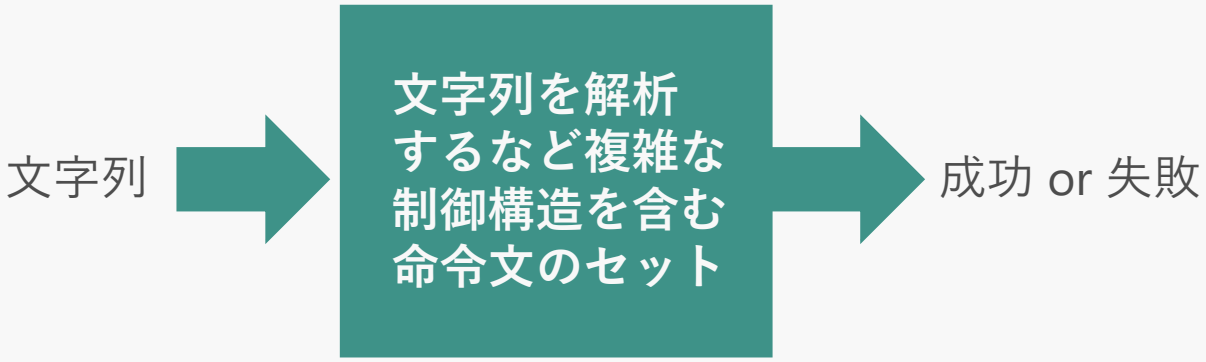
数値以外も処理できる!

数値計算の関数



z=sin(x)+y
z=max(x,y)など

複雑な処理を含む関数



文字列への返事をコンソール表示する関数
文字列をネットワーク送信する関数, など

main関数のなかがすっきりする！

関数を使わない場合

```
int main()
{
    命令文A;
    命令文B;

    命令文C;
    命令文D;
    命令文E;
    命令文F;

    命令文A;
    命令文B;
    命令文C;

    命令文D;
    命令文E;
}
```

長くて処理の流れを掴みづらい・・・

ABCDEの命令を「関数1」としてまとめよう!

ここも「関数1」にできる!

関数化した場合

```
int main()
{
    関数1;
    命令文F;
    関数1;
}
```

構造がよく分かる!

繰り返し使うような命令文のセットは関数化しておくのがよい

main関数も関数なので形は同じ!

「main関数の外」にブロックつきで記述して定義

関数の実装例

入力2値の最大値を返すmaxという名前の関数を実装している部分

```
#include <stdio.h>
int max(int x, int y){
    int z;
    if(x >= y){
        z = x;
    }else{
        z = y;
    }
    return z;
}

int main( ){
      int a = 5;
      int b = 7;
      int c = max(a, b);
}
```

入力x,y

出力z

intやdoubleなど.
返り値の型をここで設定

関数の中で入力として使う変数を必要なだけここで定義

関数の定義方法の一般形

```
返り値の型 関数名(引数の型と名, 引数の型と名...){
    命令文のセット;
    return 返り値;
}
```

return の後に書いたものが返り値になる

関数名(引数名); で呼び出せる

main関数内で関数を「呼び出す」と返り値に置き換わる

2値の最大値を求めるmax関数の実装例

```
#include <stdio.h>
int max(int x, int y){
    int z;
    if(x >= y){
        z = x;
    }else{
        z = y;
    }
    return z;
}

int main( ){
    int a = 5;
    int b = 7;
    int c = max(a, b);
}
```

関数を定義しているだけなのでmain関数内で呼び出されない限り実行はされない

Step 2
xにはaの値が，yにはbの値が入る

Step 3
zに7が入る

Step 4
7が返り値になる

Step 5
max(a,b)が返り値に変身して，cには7が代入される

Step 1
引数としてa，bを利用して関数maxを呼び出す呪文が唱えられる

関数の中で定義された変数が有効なのはブロック内だけ

2値の最大値を求めるmax関数の実装例

```
#include <stdio.h>
int max(int x, int y){
    int a;
    if(x >= y){
        a = x;
    }else{
        a = y;
    }
    return a;
}

int main( ){
    int a = 5;
    int b = 7;
    int c = max(a, b);
}
```

変数の名前は同じだが，
別世界の全く別物!!!
だまされないように!

このx,y,aの有効範囲はmax関数内

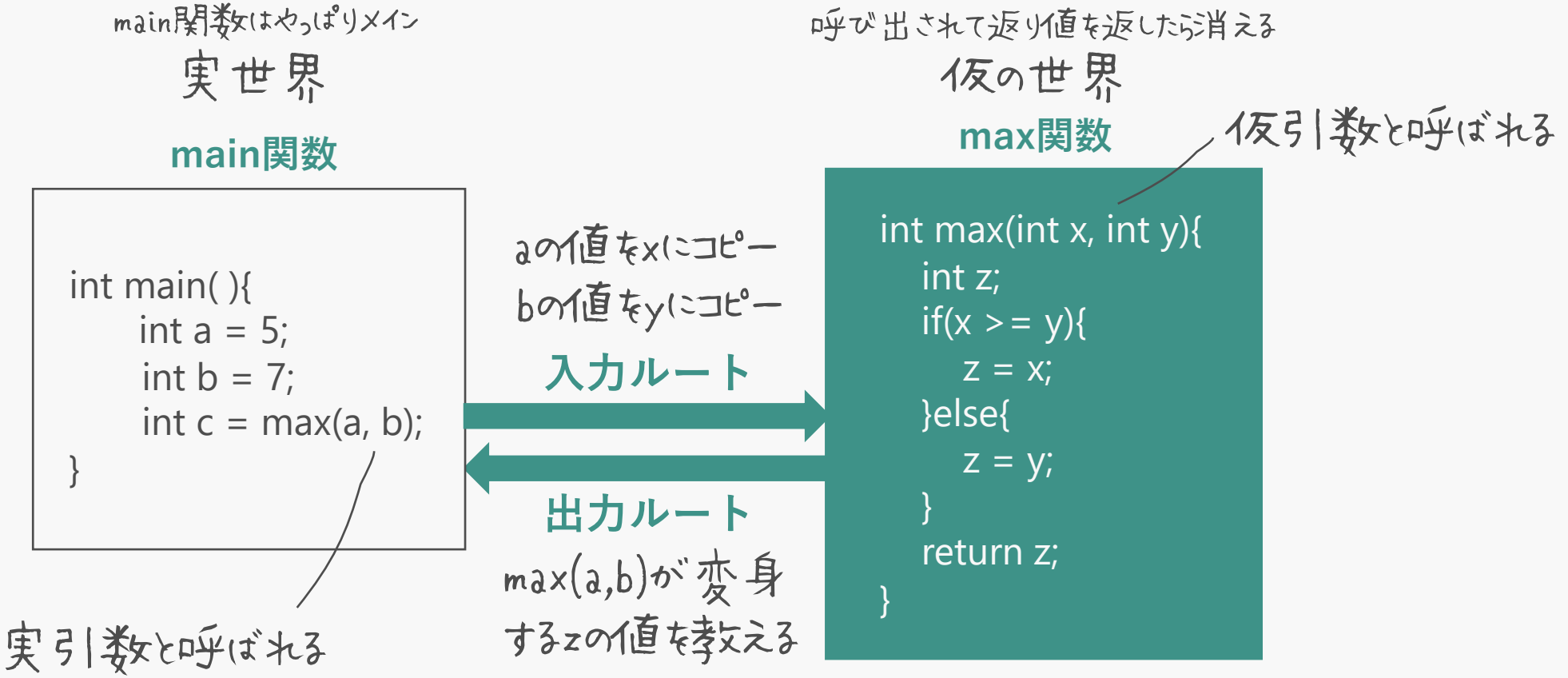
※有効範囲のことは「変数のスコープ」とも呼ばれます

このa,b,cの有効範囲はmain関数内

メリット:
他の関数の変数と名前が被っていないかをチェックせずに関数を作れるので安心

デメリット:
値を伝え合うのが面倒（基本的には引数と返り値というルートのみ）

引数と返り値という限定ルートで値をコピーして渡す



※実は「グローバル変数」という，実世界と仮の世界のどちらでも有効に存在できる変数も作れるが，いろいろな世界（関数）から干渉を受けて意図せず変えられる恐れがあるので極力使用しない

一度の関数定義でmain関数で何度も呼び出せる

2値の最大値を求めるmax関数の実装例

関数定義はmain関数前に済ませてコンパイラに教えておくのが基本.
先に教えておかないと, main関数の中でmax関数が正しく機能しない恐れがある

```
#include <stdio.h>
int max(int x, int y){
    int z;
    if(x >= y){ z = x; }else{ z = y;}
    return z;
}

int main( ){
      int a = 5;
      int b = 7;
      int c = 8;
      int d = max(a, b);
      int e = max(a, c);
}
```

スライド内での見やすさ向上のため, 改行を無くして行数削減しています (これでもコンパイル通ります)

引数を変えつつ同じ関数を使いまわせる!
これが関数の便利なところ.

必要最低限だけコンパイラに先に教えておく

「プロトタイプ宣言」を使えばソースが見やすくなる

main関数前で関数を定義している例

関数定義部

```
#include <stdio.h>
int max(int x, int y){
    int z;
    if(x >= y){ z = x; }else{ z = y;}
    return z;
}

int main( ){
    int a = 5;
    int b = 7;
    int c = 8;
    int d = max(a, b);
    int e = max(a, c);
}
```

自作の関数が増えるとmain関数が下の方に埋もれて見づらくなる

一行予告で済ませて後ろで関数を定義している例

```
#include <stdio.h>
int max(int x, int y);

int main( ){
    int a = 5;
    int b = 7;
    int c = 8;
    int d = max(a, b);
    int e = max(a, c);
}

int max(int x, int y){
    int z;
    if(x >= y){ z = x; }else{ z = y;}
    return z;
}
```

これがプロトタイプ宣言．関数定義のブロックをセミコロンに置き換えただけ

前から後ろにそのまま移動

関数が返り値に変身することを知っていれば簡単

関数の中でも関数を使える

max関数の中にmax関数を入れた例

```
#include <stdio.h>
int max(int x, int y){
    int z;
    if(x >= y){ z = x; }else{ z = y;}
    return z;
}

int main( ){
      int a = 5;
      int b = 7;
      int c = 8;
      int d = 9;
      int e = max(max(a, b),max(c, d)) ;
}
```

max(a,b)が7に変身

max(c,d)が9に変身

max(7,9)が9に変身するので
eには9が代入される

必要なければ関数の返り値は受け取らなくてもよい

printf関数の定義は
stdio.hのファイルの中に
しっかり書かれている

printf関数の２通りの使い方

```
#include <stdio.h>
int main( ){
        printf(“Hello World¥n”);
        int a =printf(“Hello World¥n”);
}
```

返り値を受け取らない使い方

返り値を受け取る使い方

実行結果

```
Hello World
Hello World
```

問題なくprintfの機能は発揮されている