C言語講習会

#2 C言語のルール

1. 前回のおさらい

ソースコード：hello.c

```c
#include <stdio.h>

int main(void){
    printf("Hello World!!¥n");
    return 0;
}
```

ターミナル(terminal)

```
$ gcc hello.c -o hello

$ ls
  hello.c  hello

$ ./hello
  Hello World!!
```

Point：print( ) のなかに “好きな文字列” を入れると画面に表示される

1. 前回のおさらい

ソースコード：hello.c

```c
#include <stdio.h>

int main(void){
    printf("Hello World!!¥n");
    return 0;
}
```

ターミナル(terminal)

```
$ gcc hello.c -o hello
$ ./hello
Hello World!!
```

プログラムは、
原則上から順に実行される

Point：print( ) のなかに “好きな文字列” を入れると画面に表示される

2. 各命令の意味

```
#include <stdio.h>
```

include【他動】：〜含める, 包含する

stdio (Standard Input Output)：標準入出力

.h : header file の拡張子

#include <stdio.h> の意味

stdio.h という名前のヘッダファイルをプログラムの中に含んでね！！

2. 各命令の意味

```
int main(void){

}
```

main( )：メイン関数
{ }の中に書いた内容がメインに実行される

int (integer)【名】：整数

void 【形】：空っぽの，中身のない

main関数の意味

準備ができたら、{ }で囲まれた部分からプログラムを実行してね！！

関数とintの意味については、後ほど説明

補足: インデント (ソースコードの読みやすさ)

インデントあり

```
int main(void) {
    printf("Hello World!!¥n");
    return 0;
}
```

インデント

インデントなし

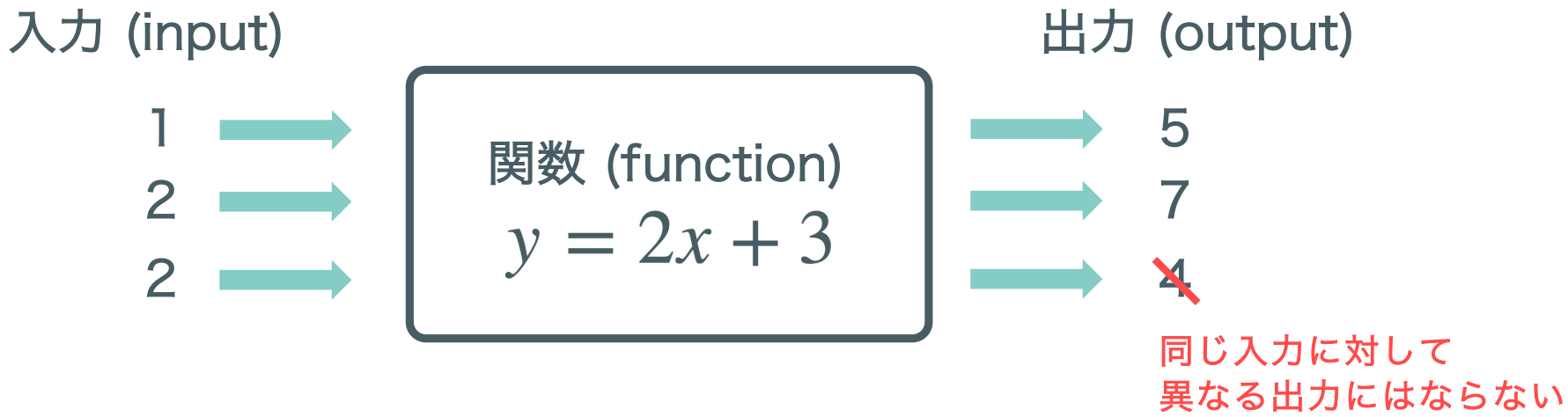
```
int main(void)
{ printf("Hello World!!¥n");
                    return 0;          }
```

インデントは Tabキー を使うと楽にできます

インデント

{ } の間に命令を記述する際に字下げすることを インデント という

補足: 関数とは



関数とは

変数x, yについて、xを入力するとyの値がただ一つに定まるとき、yをxの関数という。

補足: (数学的な)関数の表記

中学生までの表記

$$y = 2x + 3$$

$x = 2$ のとき

$y = 7$

高校生以降の表記

$$f(x) = 2x + 3$$

$f(2) = 7$

function【名】: 関数

関数の表記

$f(x)$ の変数 x に値を代入すると、結果が出力される

補足: (C言語での)関数の表記

C言語での表記

```
int main(void)
```

戻り値の型　　引数

数学での表記

$$f(x) = 2x + 3$$

プログラミング言語では引数 (ひきすう) といい、
この場合は、引数として渡すものがないので void を渡しています。

関数の表記

C言語では　戻り値の型 関数名 (引数)　で関数を定義する

2. 各命令の意味

```
printf( "Hello World!!¥n" );
```

printf : print format

printf関数の意味

引数に渡した文字列を画面に表示してね！！

stdio.hをincludeする理由

実は、printf 関数などの標準入出力を利用するには、stdio.h を読み込む必要があります。
試しに `#include <stdio.h>` をコメントアウトしてみましょう。

補足: コメントアウト

一行だけコメントアウト

```
// #include <stdio.h>
```

コメントアウトした部分は実行されません

複数行コメントアウト

```
int main(void){
/*
    printf("Hello World!!¥n");
    return 0;
*/
}
```

コメントアウト

1行だけの場合は // 、複数行の場合は /*　*/ で囲む

2. 各命令の意味

```
// #include <stdio.h>
```

実行するとエラーになる (環境によってはwarning)

```
$ gcc hello.c -o hello
hello.c:4:5: error: implicitly declaring library function 'printf' with type
'int (const char *, ...)' [-Werror,-Wimplicit-function-declaration]
    printf("Hello World!!¥n");
    ^
hello.c:4:5: note: include the header <stdio.h> or explicitly provide a
declaration for 'printf'
1 error generated.
```

2. 各命令の意味

stdio.h が保存されている場所 (Mac OS)

```
Applications/Xcode.app/Contents/Developer/Platforms/
MacOSX.platform/Developer/SDKs/MacOSX.sdk/usr/include/stdio.h
```

stdio.h (の170行目) に printf関数 が定義されている

```
int printf(const char * __restrict, ...) __printflike(1, 2);
```

stdio.h

printf関数などの標準入力関数がたくさん定義されているファイル

2. 各命令の意味

```
return 0;
```

return【他動】：〔~を〕返す、戻す、返上する

return 0; の意味

関数を終了し、外側に戻り値(0)を返す

`return 0;` はmain関数の最後に記述しましたが、`return 0;` の後に `printf("add message¥n");` を追加するとどうなるか実行してみましょう

2. 各命令の意味

```c
#include <stdio.h>

int main(void){
    printf("Hello World!!¥n");
    return 0;
    printf("add message¥n");
}
```

```
$ gcc hello.c -o hello

$ ./hello
  Hello World!!
```

↑add message が表示されない

return 0; で main関数を終了するため、add message が表示されません

補足: 値の型

C言語では整数や文字など、値の種類によって ”型” が決まっています

型の名称	読み方	意味	値の例
int	イント	整数型	0, 2, 19, など
double	ダブル	倍精度浮動 小数点数型	0.2900, 7.3450, など
char	キャラ(チャー)	文字型	‘a’, ‘x’, ‘z’, など
void	ボイド	空のデータ型	

詳しい内容は、“変数” について講義する際に再度説明します。

2. 各命令の意味

```
int  main(void){
    return 0 ;
}
```

main関数の戻り値は 0 (int型) なので
`int main(void);` となる。

もちろん、戻り値がない関数であれば、
`void func(void);` となる。

関数の表記 (復習)

C言語では　戻り値の型 関数名 (引数)　で関数を定義する

参考文献

・大川内隆郎, 大原竜男, かんたん C言語 [改訂2版], 技術評論社, 2017.

・筧捷彦, 高田大二 他, 入門 C 言語, 実教出版株式会社, 2019.