

# 知能情報工学演習I 第7回(後半第1回)

岩村雅一

[masa@cs.osakafu-u.ac.jp](mailto:masa@cs.osakafu-u.ac.jp)



# 後半の予定

7. 5月31日 プログラミング環境(テキスト1,2章)
8. 6月7日\* 変数とデータ型(3章)、演算子(4章)
9. 6月14日 コンソール入出力(6章)、配列(3章)
10. 6月21日\* 制御文1(テキスト5章)
11. 6月28日\* 制御文2(テキスト5章)
12. 7月12日 関数1(テキスト7章)
13. 7月19日 配列(3章)、応用プログラム



# 授業のウェブページ

- 授業で用いたプレゼン資料や課題はウェブページで公開する

<http://www.m.cs.osakafu-u.ac.jp/~masa/class/>



# 本日のメニュー

- プログラム作成→実行
- Cのプログラムの基本
- 文字列の表示
- 変数
- コンパイラの使い方



# プログラム作成→実行の流れ

1. プログラムの作成
2. コンパイル
3. 実行

# 1. プログラムの作成

- 次のプログラムを打ち込んで、smp1.cという名前で保存しましょう(テキストのP.22)

```
#include <stdio.h>
    (改行だけ入力する)
int main(void)
{
    printf("Hello¥n");
    (改行だけ入力する)
    return 0;
} (空白を入れる)
```



## 2. コンパイル

- C言語のプログラムはそのままでは実行することができない
  - ➡ コンピュータが実行できるように変換が必要
  - ➡ コンパイラ
- この授業ではgccと呼ばれるコンパイラを利用
  - Windowsでも、UNIXともにフリーで使える



## 2. コンパイル

## 3. 実行

- 打ち込んだプログラム(smp1.c)をgccでコンパイルする(テキストのP.11)
  1. gcc smp1.cと打ち、コンパイルする
  2. lsでa.outができていることを確認する
  3. a.outと打ってプログラムを実行する



# Cのプログラムの基本1

## ■ プログラム

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

← 用意された関数(標準関数)を使うときに必要

```
int main(void)
```

← main関数は必ず作成する

```
{  
    printf("Hello¥n");
```

{}で囲まれた部分が関数の中身

```
    return 0;
```

← main関数の最後に付ける

```
}
```

# Cのプログラムの基本2

## ■ プログラム

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

```
int main(void)
```

```
{
```

```
    printf("Hello¥n"); ← 文は「;」で終わる
```

```
    return 0; ← 文は「;」で終わる
```

```
}
```

# 文字列の表示

## ■ プログラム

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

```
int main(void)
```

```
{
```

```
printf("Hello¥n");
```

文字列を表示

文字列

```
return 0;
```

```
}
```

# 文字列の表示

## ■ 文字列

□ `printf("Hello¥n");`

H	e	l	l	o	¥n
---	---	---	---	---	----

↖ 改行

□ `printf("Hi!¥nHow are you?¥n");`

H	i	!	¥n	H	o	w		a	r	e		y	o	u	?	¥n
---	---	---	----	---	---	---	--	---	---	---	--	---	---	---	---	----

↖ 改行

↖ 改行



# 変数

- プログラム内で数字や文字を記憶するため、**変数**が利用される
  - 変数は覚える値の種類によって異なる
    - ➡ いくつかの種類(**型**)が用意されている
  - 使う変数は最初に**宣言**しなくてはならない

# サンプルプログラム

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

```
int main(void){
```

```
char a1, a2, a3;  
int b1, b2, b3;
```

```
a1 = 6;
```

```
a2 = 2;
```

```
b1 = 192;
```

```
b2 = 168;
```

変数

```
a3 = a1 + a2;
```

```
b3 = b1 + b2;
```

```
printf("a1=%d a2=%d\n", a1, a2);
```

```
printf("b1=%d b2=%d\n", b1, b2);
```

```
printf("a1+a2=%d\n", a3);
```

```
printf("b1+b2=%d\n", b3);
```


```
return(0);
```

```
}
```



# 変数の解説

- char, intは整数を覚える
- 小数を覚える変数もある
- charとintは覚えられる数の大きさに違いがある(テキストP.51)
  - char: -128~127
  - int: -2147483648~ 2147483647



# 変数の名前の付け方のルール (テキストP.37)

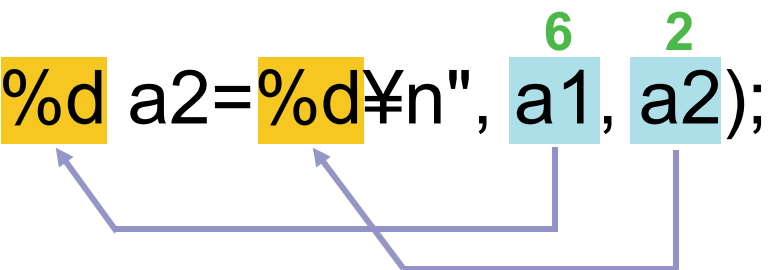
- 名前に使える文字
  - アルファベット(大文字、小文字)
  - 数字
  - アンダーバー(\_)
- 名前の最初はアルファベットかアンダーバー
- 名前の長さは31文字まで
- 予約語(あらかじめCで使う名前)は使用不可
- 大文字と小文字は区別される



# 文字列の表示

## ■ 文字列

□ `printf("a1=%d a2=%d¥n", a1, a2);`



□ 出力


a	1	=	6		a	2	=	2	¥n
---	---	---	---	--	---	---	---	---	----

↖ 改行



# 実行ファイルの名前を変える

- gcc (Cのプログラム)とすると実行ファイルが a.outになってしまう
  - 実行ファイル名を変えたいとき
    - gcc -o (実行ファイル名) (Cのプログラム)
  - gcc -o smp1 smp1.cとしてコンパイルしてみよう
    - 実行ファイルがsmp1としてできるので、実行してみよう



# gccを使う上での注意点

- Cのプログラムは.cとしなくてはいけない



# プログラムの間違い

## ■ プログラムの間違いには大きく2つある

### □ 文法の違い


- 文字の打ち間違い
- 関数の使い方が間違っている

⇒ コンパイラがエラー/警告を出す

### □ 計算方法の違い

- 問題を解く解法が間違っている(例:「5+3」→「5-3」)
- 希望の動作をするようプログラムが書かれていない

⇒ プログラムをテストして判断する



# コンパイラのエラー/警告

- エラーは直さないとプログラムが動かない
- 警告は無視してもプログラムは動作する  
ただし、なぜ警告が出たかは把握すること



## エラーの例

- printfをprintとした場合  
smp1.c: undefined reference to `print'
- mainをmaniと打ち間違えた場合  
Undefined reference to `main'
- printf()の“を忘れた場合  
smp1.c: In function `main':  
smp1.c:5: `Hello' undeclared (first use in  
function)  
.....



# 課題

- 変数 $a$ にあながた生まれた月、変数 $b$ に生まれた日を代入して、 $a+b$ と $a-b$ を計算して出力するプログラムを作成しなさい
- ただし、レポートはLaTeXで作成し、DVIファイル(.dvi)とCのソース(.c)を提出すること
  - レポートには名前、学籍番号、授業に対するコメント(任意)を書く
  - Cのソースは`¥begin{verbatim}`と`¥end{verbatim}`で囲む(5月10日の資料の6.1を参照)