

# 2019年 C言語講座

~第1回 基礎の基礎編~

# 目次

1. プログラムの書き方と表示
2. 変数
3. 読み込みと表示

# 目次

1. プログラムの書き方と表示

2. 変数

3. 読み込みと表示

# 1. プログラムの書き方と表示

## ▶ プログラムの作成から実行までの流れ

### 1. プログラムを書く



めっちゃ凄いソースプログラム書けたで！！

**ソースプログラム** : “文字の並び”として作ったプログラム

\* **ソースファイル** : ソースプログラムを格納したファイル

# 1. プログラムの書き方と表示

## ▶ プログラムの作成から実行までの流れ



```
hello.c
保存(S)

/*挨拶するプログラム*/
#include<stdio.h>

int main(){

    //文字の表示
    printf("Hello!!\n");

    return 0;
}

C ▾ タブ幅: 4 ▾ (11行、1列) ▾ [挿入]
```

# 1. プログラムの書き方と表示

▶ プログラムの作成から実行までの流れ

2. パソコンが理解できる言葉に翻訳

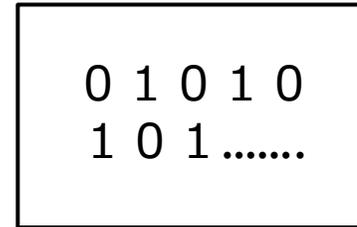


コンパイル (翻訳)



gcc -o hello hello.c

実行プログラム



```
hello.c
/*挨拶するプログラム*/
#include<stdio.h>
int main(){
    //文字の表示
    printf("Hello!\n");
    return 0;
}
```



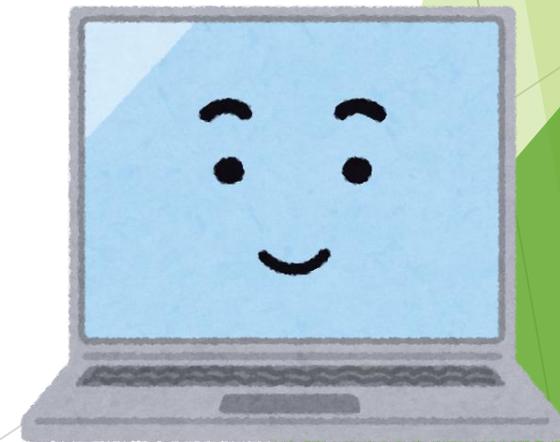
# 1. プログラムの書き方と表示

▶ プログラムの作成から実行までの流れ

3. 実行する



これ実行して

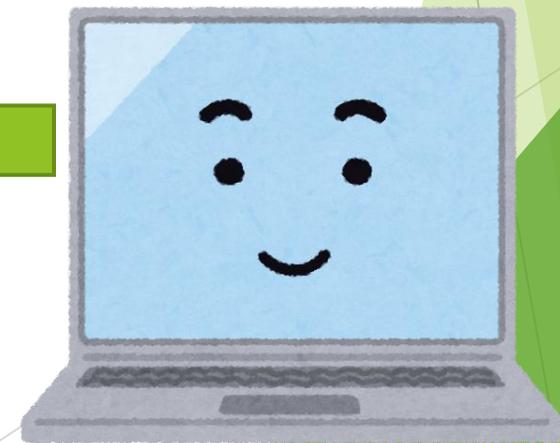


# 1. プログラムの書き方と表示

▶ プログラムの作成から実行までの流れ

3. 実行する

実行



```
op@op: ~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
op@op:~$ gedit hello.c
op@op:~$ gcc -o hello hello.c
op@op:~$ ./hello
Hello!!
op@op:~$
```

実際にやってみよう！！

# 1. プログラムの書き方と表示

## 1. プログラムを書く

テキストエディタ

geditやemacs,viなどのこと

他にも色々種類があるので自分に合ったもの  
を見つけておこう



宗教上の理由で本講座では基本的にgeditを使ってもらう

# 1. プログラムの書き方と表示

## 1. プログラムを書く

下のプログラムをメモ帳アプリ（gedit）に書こう

```
#include <stdio.h>

int main(void)
{
    printf(“こんにちは”);

    return 0;
}
```

\*C言語のソースファイルには必ず.cとつける



書いたら hello.c という名前で保存

# 1. プログラムの書き方と表示

## 2. コンパイル (今回の場合)

```
gcc -o hello hello.c
```

- 解説

- -o : 付けて実行すると実行ファイルに名前をつけられる
- hello : 実行ファイルにつける名前
- hello .c : 実行ファイルの元となるソースファイル



今後は必要に応じて各部分を変えてやる

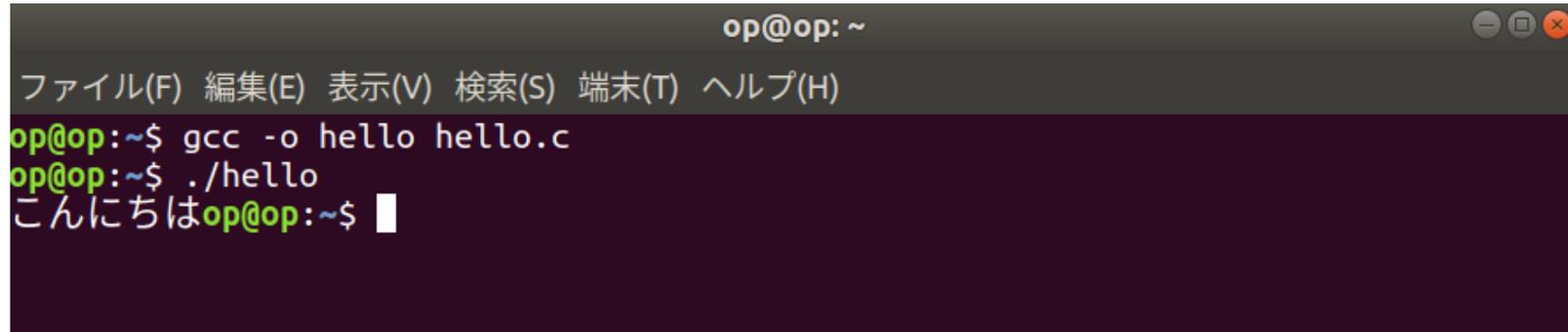
# 1. プログラムの書き方と表示

## 3. 実行

`./hello`

プログラムの実行には、`./`の後ろに実行ファイル名

- 実行結果例



```
op@op: ~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
op@op:~$ gcc -o hello hello.c  
op@op:~$ ./hello  
こんにちはop@op:~$
```

# 細かいところの解説

# 1. プログラムの書き方と表示

## ▶ 細かいところ

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

左の赤文字部分は“おまじない”として覚える

```
int main(void)
```

```
{
```

```
    printf(“こんにちは”);
```



意味は後々わかる

```
    return 0;
```

```
}
```

# 1. プログラムの書き方と表示

## ▶ printf関数

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

```
int main(void)
```

```
{
```

```
    printf(“こんにちは”);
```

```
    return 0;
```

```
}
```

```
printf(“文章など”);
```

- “”の中身を表示する。
- “こんにちは¥n”とすると改行できる



```
op@op: ~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
op@op:~$ gcc -o hello hello.c
op@op:~$ ./hello
こんにちは
op@op:~$
```

\* C言語では基本的に命令の後ろには ; をつける

# 1. プログラムの書き方と表示

## ▶ printf関数

### 問1.1

hello.c を改変し,次のように表示せよ

```
op@op: ~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
op@op:~$ gcc -o hello hello.c
op@op:~$ ./hello
こんにちは
op@op:~$ gcc -o hello hello.c
op@op:~$ ./hello
おはよう
こんにちは
こんばんは
op@op:~$
```

# 1. プログラムの書き方と表示

## ▶ printf関数

問1.1  
解答)

```
#include <stdio.h>

int main(void)
{
    printf(“おはよう¥nこんにちは¥nこんばんは¥n”);

    return 0;
}
```

# 1. プログラムの書き方と表示

▶ printf関数で数字や計算結果の表示

```
printf("%d", 15 + 37);
```

- ()の中身を**実引数**という。この場合, "%d"と15 + 37のこと。
- 実引数が2つ以上ある場合 **,** で区切る
- "%d"は, 続いて与える実引数の値を**10進数**で表示するという意味



実行すると52と表示される

# 1. プログラムの書き方と表示

- ▶ printf関数で数字や計算結果の表示（確認）

以下の文を書き写そう

```
/* math.c */
#include <stdio.h>
int main(void)
{
    //計算結果の表示
    printf("15と37の和は%dです。¥n", 15 + 37);

    return 0;
}
```



```
op@op: ~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
op@op:~$ gedit math.c
op@op:~$ gcc -o math math.c
op@op:~$ ./math
15と37の和は52です。
op@op:~$
```

# 1. プログラムの書き方と表示

## ▶ コメント（注釈）

```
/*  
    math.c  
*/  
#include <stdio.h>  
int main(void)  
{  
    //計算結果の表示  
    printf("15と37の和は%dです。¥n", 15 + 37);  
  
    return 0;  
}
```

赤字の部分を**コメント**という。  
複数行なら**/\*\*/**で囲い、一行のみなら**//**を使う。

動作についてのメモを残す時などに使う。

# 目次

1. プログラムの書き方と表示

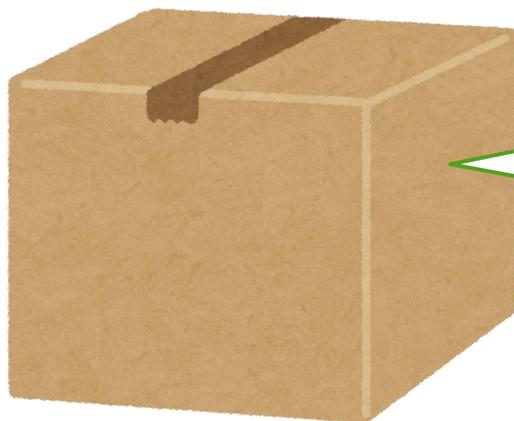
2. 変数

3. 読み込みと表示

## 2. 変数

### ▶ 変数とは

数字や文字を格納する為の箱のこと



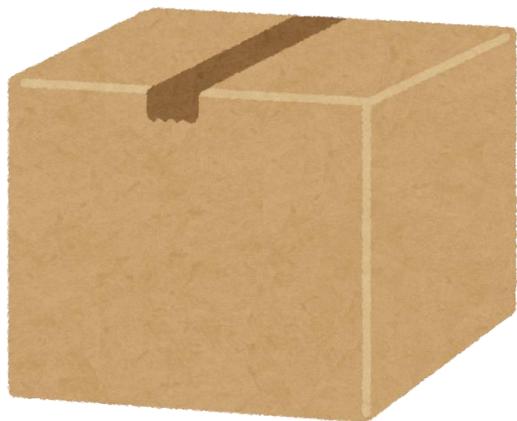
- ・ 結果を保存したり,後で使う値を入れておける。
- ・ 中身を自由に変更できる。

## 2. 変数

### ▶ 変数の作り方

まず宣言する

```
int n;
```



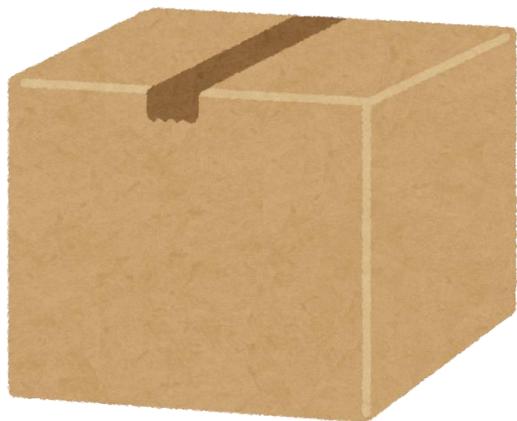
名前 : n

## 2. 変数

### ▶ 変数の作り方

まず宣言する

```
int n;
```



名前 : n

- 「nという名前のint型の変数作る」という意味。
- int型は整数のみ格納できる。

## 2. 変数

### ▶ 変数の使い方

以下の文を書き写そう

```
#include <stdio.h> /* variable.c */
int main(void)
{
    int x, y;

    x = 21;
    y = x + 9;

    printf("xの値は%dです。¥n", x);
    printf("yの値は%dです。¥n", y);
    return 0;
}
```

• =は「**右の値を左の変数に代入する**」という意味。

• =には左右が**等しい**という意味はない

## 2. 変数

### ▶ 変数の使い方(結果)

```
op@op: ~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
op@op:~$ gedit variable.c&  
[1] 22971  
op@op:~$ gcc -o variable variable.c  
op@op:~$ ./variable  
xの値は21です。  
yの値は30です。  
op@op:~$
```

## 2. 変数

### ▶ 変数の使い方の演習

#### 問2.1

計算結果の表示（スライド20）で作成したmath.cについて、printf内に数字を一切使わず同様の結果を得られるように改変せよ。

```
op@op: ~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
op@op:~$ gedit math.c  
op@op:~$ gcc -o math math.c  
op@op:~$ ./math  
15と37の和は52です。  
op@op:~$
```

ヒント.

- 変数を3つ使う。

## 2. 変数

### ▶ 変数の使い方の演習

問2.1

解答例)

```
#include <stdio.h>
int main(void)
{
    int x, y , z ;

    x = 15
    y = 37
    z = x + y

    printf(“%dと%dの和は%dです。 ¥n”, x, y, z);

    return 0;
}
```

## 2. 変数

### ▶ 変数の使い方の演習

問2.1  
別解)

```
#include <stdio.h>
int main(void)
{
    int x = 15, y = 37, z = x + y;

    printf(“%dと%dの和は%dです。 ¥n”, x, y, z);

    return 0;
}
```

- 宣言と同時に代入しておく。これを初期化という。
- この時の=は初期化子という。

## 2. 変数

### ▶ 初期化の必要性

問2.1のソースプログラムを次のように変更して実行してみよう

```
#include <stdio.h>
int main(void)
{
    int x, y, z ;

    printf(“%dと%dの和は%dです。¥n”, x, y, z);

    return 0;
}
```



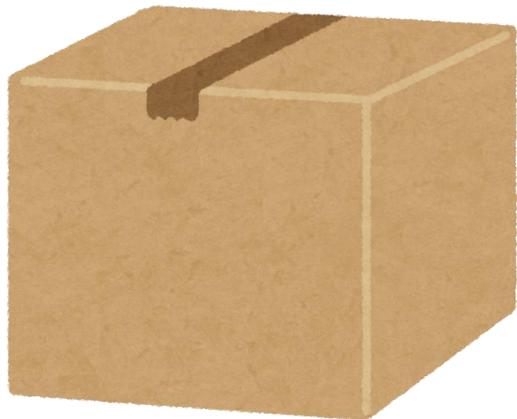
滅茶苦茶な値が表示される

## 2. 変数

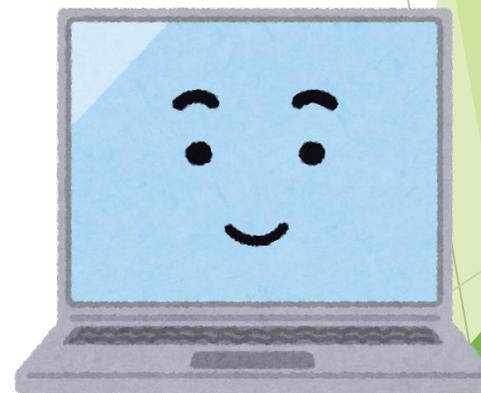
### ▶ 初期化の必要性

何故か？

```
int x;
```



名前：x

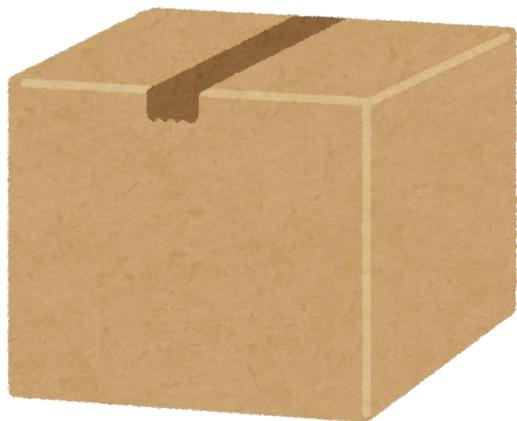


## 2. 変数

### ▶ 初期化の必要性

何故か？

```
int x;
```



名前：x

不定な値を代入



変数できたけど,初期化の命令が無い...

適当に入れるかー。



## 2. 変数

### ▶ 初期化の必要性



この結果変な値が表示される. . .

これを防ぐためにも特に不要でない限り初期化をしておく！！

# 目次

1. プログラムの書き方と表示

2. 変数

3. 読み込みと表示

## 3. 読み込みと表示

### ▶ キーボードからの読み込み

ただ表示するだけでなく,キーボードから何か入力したい



**scanf関数**を使う

## 3. 読み込みと表示

### ▶ scanf関数

```
scanf(“%d”, &n);
```

- int型の変数nに, キーボードから入力された整数値が代入される。
- &が必要なことに注意！！

## 3. 読み込みと表示

### ▶ scanf関数 (練習)

```
#include <stdio.h> /*scanf_prac.c*/
Int main(void)
{
    int x;

    printf(“整数を入力してください: ”);
    scanf(“%d”, &x);

    printf(“あなたが入力したのは%dです。¥n”, x);

    return 0;
}
```

## 3. 読み込みと表示

### ▶ scanf関数 (練習)

#### 実行結果

```
op@op: ~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
op@op:~$ gcc -o scanf_prac scanf_prac.c
op@op:~$ ./scanf_prac
整数を入力して下さい:35
あなたが入力したのは35です。
op@op:~$ █
```

## 3. 読み込みと表示

### ▶ scanf関数 (演習)

#### 問3.1

2つの整数を入力してもらいその和を表示するようにscanf\_prac.cを  
改変せよ。

```
op@op: ~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
op@op:~$ gcc -o scanf_prac scanf_prac.c
op@op:~$ ./scanf_prac
整数xを入力して下さい:4
整数yを入力して下さい:9
xとyの和は13です。
op@op:~$ █
```

## 3. 読み込みと表示

### ▶ scanf関数 (演習)

問3.1  
解答)

```
#include <stdio.h>
int main(void)
{
    int x,y;

    printf(“整数xを入力してください: ”);
    scanf(“%d”, &x);

    printf(“整数yを入力してください: ”);
    scanf(“%d”, &y);

    printf(“xとyの和は%dです。 ¥n”, x+y);

    return 0;
}
```

## 3. 読み込みと表示

### ▶ puts

```
puts(“文字”);
```

- \nを書かなくても改行して表示してくれる。
- ただし,%dなどが使えない

## 3. 読み込みと表示

### ▶ puts (練習)

```
#include <stdio.h>
int main(void)
{
    int x = 37;

    puts(“整数xは37です。”);

    puts(“整数xは%dです。”,x);

    return 0;
}
```

これをコンパイルしようとする...

## 3. 読み込みと表示

### ▶ puts (練習)

```
op@op: ~
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
op@op:~$ gcc -o puts_prac puts_prac.c
puts_prac.c: In function 'main':
puts_prac.c:9:2: error: too many arguments to function 'puts'
  puts("整数xは%dです",x);
  ^~~~
In file included from puts_prac.c:1:0:
/usr/include/stdio.h:632:12: note: declared here
extern int puts (const char *__s);
           ^~~~
op@op:~$
```

putsでは%dで変数の中身を表示できない！！

## 3. 読み込みと表示

### ▶ puts (演習)

#### 問3.2

文字の表示にputsのみを使用して次のような結果が得られるプログラムを書け。



```
op@op: ~  
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)  
op@op:~$ ./puts_prac  
中  
華  
統  
一  
op@op:~$
```

## 3. 読み込みと表示

### ▶ puts (演習)

#### 問3.2 解答)

```
#include <stdio.h>
int main(void)
{
    puts(“中”);
    puts(“華”);
    puts(“統”);
    puts(“一”);

    return 0;
}
```

今回の内容は以上です。