

C言語入門

第2回



本日の流れ

- 前回までの復習
- 宿題(自己紹介)の見せ合い
- 構文のお勉強
 - if文の続き
 - switch文
 - for文
 - while文

復習

前回までの復習

復習

前回の内容

- C言語とは
 - コンパイル
 - 関数
 - printf関数
 - 計算
 - 変数
 - if文
- プログラミング言語
機械語へ翻訳
まとまった仕事を行う単位
文字列の表示
 $5 \times 3 \rightarrow 5 * 3$
a,bを使って計算
もし雨が降ったら・・・

復習

C言語とは

- シンプルでわかりやすいプログラミング言語
- プログラムの指示通りにコンピュータを動かすことができる

```
#include <stdio.h>

int main(void){

    printf("Hello, world\n");
    return 0;
}
```



Hello, world

復習

コンパイル

プログラム

printf, if などの文字列

C言語のイメージ

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

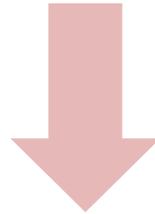
```
Int main(void)
```

```
{
```

```
    printf("Hello");
```

```
    return 0;
```

```
}
```



コンパイル(翻訳)

機械語

2進数(0と1だけ)で動く

機械語のイメージ

```
0001010011010101
```

```
0001010101010100
```

```
1010100010101011
```

```
0001010010101010
```

```
0001010010101010
```

関数

- データを受け取り、定められた通りの処理を実行して結果を返す一連の命令群

二つの数(引数)を受け取って
それらを合計した値を返す



2と3を与える

5という結果が返ってくる

好きなときに
使える!

- 多くの言語では、よく使う機能が関数としてあらかじめ用意されている

復習

おまじない部分

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

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

```
    printf("Hello, world¥n");  
    return 0;
```

```
}
```

} main関数

⇒ main関数の中に書いていく

復習

printf関数

- 文字列を表示させる
- printf(“文字列”);

重要!!

¥n ...改行

```
#include <stdio.h>

int main(void){

    printf(“Hello, world¥n”);
    return 0;
}
```



Hello, world

復習

数値と文字列は全く違う

プログラム

- printf("5 + 4");
 - printf("%d", 5 + 4);
- 

%d...そこに整数を表示する

- printf("文字列");
 - printf("%d", 引数);
- 

実行した結果

5 + 4
9

文字列
引数が渡され表示

データの入れ物 「変数」

変数とは

[**データ型**] [**変数名**] ;

の形で宣言される、数値や文字を入れておく入れ物です。

データの型	種類	書式
int	整数	%d
double	浮動小数点	%lf
char	文字定数	%c

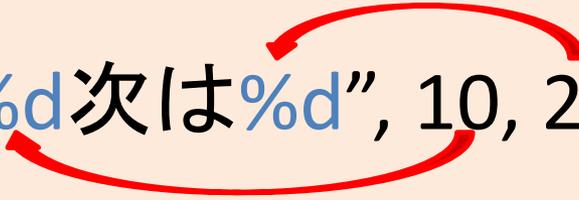


復習

%dの使い方

- 複数の数値を文字列に表示したい
 - 表示したい場所に%d
 - 引数は「,」で区切る

```
printf(“最初は%d次は%d”, 10, 20);
```



最初は10次は20

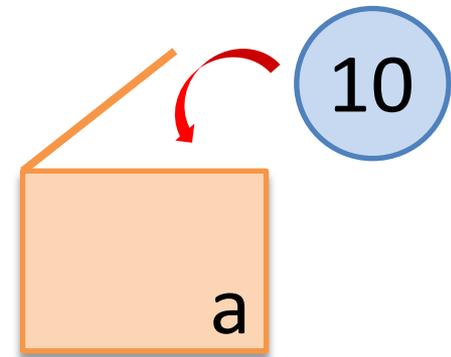


変数aのイメージ

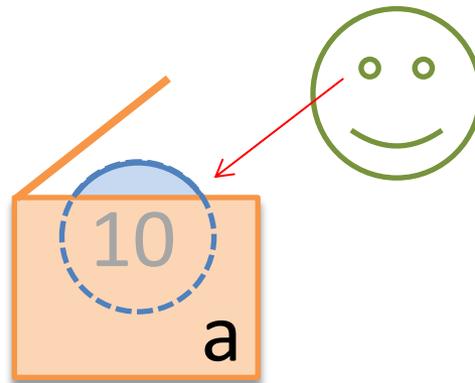
1. aという箱を作る(定義)



2. 箱に値を入れる(代入)



3. 値を見る



復習

変数は準備が必要

- `int a;`
 - `a = 10;`
- aはint(整数)型の変数(定義)
aに10を代入(初期化)

⇒ 変数aが使える

- `printf("%d", a * a);`
- 引数

重要!!

C言語で「=」は代入
aは10と等しい

```
a = 10;  
a == 10;
```

変数の名前

- x, aだけでなく sum, ageなど自由
- 半角英数、「_(下線)」 (Aとaは異なる文字)
- 先頭は非数字でなければならない(2012yearは×)
- プログラムの意図が伝わるような名前を！

{ sum : 合計値
age : 年齢

```
int sum;  
sum = 2 + 3;  
printf(“%d”, sum);
```

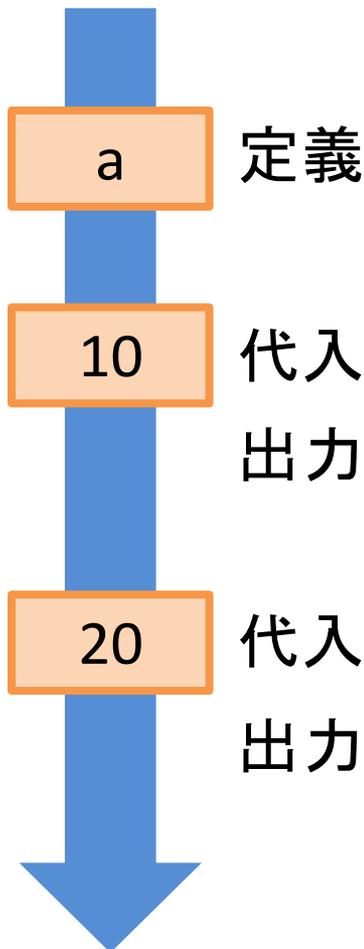
```
int age;  
age = 20;  
printf(“太郎は%d歳です。”, age);
```

復習

プログラムは上から読む

プログラム

```
int a;  
  
a = 10;  
printf(“%d¥n”, a);  
  
a = 20;  
printf(“%d¥n”, a);
```



実行した結果

```
10  
20
```

宿題(自己紹介)の見せ合い

画面が見づらいので拡大

- Windowsボタン(左下) と 「+」ボタン(右)



- Windows と Escボタン(左上)

if文

もしも・・・ならば

if文の構造

- もしも・・・ならば

```
if (条件) {  
    条件が成り立つときの処理;  
}
```

😊 条件に当てはまらないものは
どうなるのだろうか？

➡ 何もしない

if else文

```
if (条件) {  
    条件が成り立つときの処理  
}
```

```
else {  
    条件が成り立たないときの処理  
}
```

条件に入らない場合について
ちゃんと処理を考えていますよ！

セットで覚えよう

if else文

```
if (条件) {  
    条件が成り立つときの処理  
}
```

```
else if(条件2){  
    条件2が成り立つ時の処理2  
}
```

```
else{  
    条件が成り立たないときの処理  
}
```

前回の例

```
if (n >= 50){  
    1～10まで足す;  
}  
else{  
    printf(“傘を持っていかない¥n”);  
}
```

n = 1500 と入力されていたら？

実行結果



傘をもっていく

降水確率1500%？

親切なプログラム



```
if (n < 0 || 100 < n){  
    printf("降水確率は0~100にしてください");  
}  
else if (n >= 50){  
    printf("傘を持っていく¥n");  
}  
else{  
    printf("傘を持っていかない¥n");  
}
```

	または
&&	かつ

条件

$a > 10$	aは10より大きい
$a < 10$	aは10より小さい
$a \geq 10$	aは10以上(10を含む)
$a \leq 10$	aは10以下(10を含む)
$a == 10$	aは10と等しい
$a != 10$	aは10と等しくない

	または
&&	かつ

switch文

あれか、これか、それとも・・・
(多くの選択肢から1つ選ぶ)

switch文とは

if文の復習

```
if (n == 1){
    printf("あか¥n");
}
else if (n == 2){
    printf("あお¥n");
}
else if (n == 3){
    printf("みどり");
}
else {
    printf("しろ");
}
```

なんとなく見づらい...



```
switch (n){
case 1:
    printf("あか¥n");
    break;
case 2:
    printf("あお");
    break;
case 3:
    printf("みどり");
    break;
default:
    printf("しろ");
    break;
}
```

switchの構造

- 多くの選択肢から1つ選ぶ

```
switch(式){
```

```
case 定数式1:  
    処理1;  
    break;
```

```
...
```

```
default:  
    処理3;  
    break;
```

```
}
```

```
switch (n){
```

```
case 1:
```

```
    printf("あか¥n");
```

```
    break; これが処理の終わり
```

```
    省略
```

```
default:
```

```
    printf("しろ");
```

```
    break;
```

```
}
```

switch文の例

```
int c;  
c = 1;  
switch(c){  
case 1:  
case 2:  
    printf("あか¥n");  
    break;  
case 3:  
    printf("あお¥n");  
    break;  
default:  
    printf("しろ¥n");  
    break;  
}
```

for文

繰り返し処理(～の間)

for文とは

```
printf("Hello, world\n");  
printf("Hello, world\n");
```



```
int i;  
  
for(i = 1; i <= 10; i = i + 1){  
    printf("Hello, world\n");  
}
```

繰り返す処理を
まとめる!!

とても見づらい...

for文の構造

- 繰り返し処理(決まった回数)

```
for (初期化; 条件; 次の一歩){  
    繰り返す処理;  
}
```

- 初期化・・・最初に一度だけ実行される
- 条件・・・満たされている間繰り返しを続ける
- 次の一歩・・・繰り返すごとにいつも実行される

```
for (初期化; 条件; 次の一歩){  
    繰り返す処理;  
}
```

変数*i*の定義
(カウント変数)

$i = i + 1$

```
int i;  
    初期化   条件   次の一歩  
for(i = 1; i <= 10; i = i + 1){  
    printf("Hello, world¥n");  
}
```

一歩ずつ
考えてみよう!

- 初期化・・・最初に一度だけ実行される
- 条件・・・満たされている間繰り返しを続ける
- 次の一歩・・・繰り返すごとにいつも実行される

for文の例

```
int i;  
int sum;  
  
sum = 0;  
for(i = 1; i <= 10; i = i + 1){  
    sum = sum + 1;  
}    printf(“%d¥n”, sum);  
printf(“%d¥n”, sum);
```



```
10  
2  
3  
...  
8  
9  
10
```

問題

- 1～10までの数を足すプログラムを作ってみよう
 - forループ(変数i)を使う
 - 変数sumに計算結果をしまう
 - 最後に、計算結果sumをprintfで表示させる