

# C言語講座

## 第3回

繰り返し—LOOP—

# 繰り返すということ

通常、プログラムは下へ下へと流れるように実行される。



- 1
- 2
- 3

```
#include <stdio.h>
void main(void)
{
    int number;
    scanf("%d", &number);
    printf("入力結果 : %d¥n", number);
}
```

なら、printfを20回行いたいなんて時は、  
printf(“%d”, 1)を20回も書かなきゃいけない  
のだろうか？



1

2

20

```
#include <stdio.h>
void main(void)
{
    printf("%d", 1);
    printf("%d", 1);
    ...
    printf("%d", 1);
}
```

# 繰り返し①(do-while)

そんな面倒を減らしてくれるのが繰り返し文。  
似たような処理を何度でも呼び出すことが出来る。

## ○構文

```
do {  
    処理;    //条件を満たす間、繰り返す  
} while (条件);
```

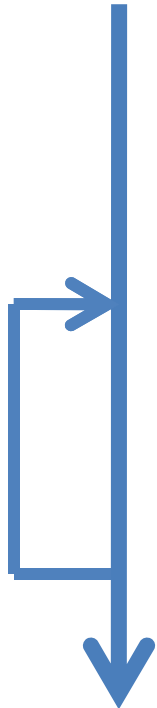
## ○プログラミング実践

### 1から5の総和を求める

```
#include <stdio.h>
void main(void)
{
    int num = 1;
    int sum = 0;
    do {
        sum += num;
        num++;
    } while (num <= 5);
    printf("%d¥n", sum);
}
```

## ○解説

```
#include <stdio.h>
void main(void)
{
    int num = 1;    // 足し合わせる数
    int sum = 0;    // 総和
    do {
        sum += num;
        num++;
    } while (num <= 5);
    printf("%d\n", sum);
}
```



## 繰り返し②(while)

条件分岐の時、ifとswitchがあったみたいに、繰り返し文にも何種類かの書き方がある。

### ○構文

```
while (条件) {  
    処理;    //条件を満たす間、繰り返す  
}
```

## ○プログラミング実践

入力された数から0までカウントダウンする

```
#include <stdio.h>
void main(void)
{
    int num ;
    scanf("%d", &num);
    while (num >= 0) {
        printf("%d ", num);
        num--;
    }
}
```



# do-whileとwhileの違い

下の二つを実行して、実際に確かめてみると...

○do-while文

```
#include <stdio.h>
void main(void)
{
    int num= 0;
    do {
        printf("do-while¥n");
    } while (num == 1);
}
```

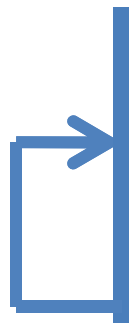
○while文

```
#include <stdio.h>
void main(void)
{
    int num= 0;
    while (num == 1) {
        printf("while¥n");
    }
}
```

# 強制中断①(break)

例えばwhile(1 == 1)なんて書いたら、永遠に終わらないプログラムになってしまう。

そこで、ループの中から強制的に抜けることが出来る処理がある。



```
#include <stdio.h>
void main(void)
{
    while (1 == 1) {
        printf("while¥n");
    }
}
```

## ○プログラミング実践

### 0から9までカウントアップする

```
#include <stdio.h>
void main(void)
{
    int num = 0;
    while ( 1 == 1 ) {
        if (num >= 10) { break; }
        printf("%d ", num);
        num += 1;
    }
}
```

## 強制中断②(continue)

continueは「継続する」なんて意味がある。

breakは完全にループ処理を抜けるのに対し、continueは一時的に処理を抜かすために使う。

文字だけでは分からないと思うので次のプログラム実践を見て熟考してほしい。

## ○プログラミング実践

### 0から9までの偶数を表示する

```
#include <stdio.h>
void main(void)
{
    int num = 0;
    while ( num < 9 ) {
        num += 1;
        if (num % 2 == 1) { continue; }
        printf("%d ", num);
    }
}
```

# breakとcontinueの違い

下の二つを実行して、実際に確かめてみると...

○break文

```
#include <stdio.h>
void main(void)
{
    int num= 0;
    while (num++ < 3) {
        if (num == 1) { break; }
        printf("%d¥n", num);
    }
}
```

○continue文

```
#include <stdio.h>
void main(void)
{
    int num= 0;
    while (num++ < 3) {
        if (num == 1) { continue; }
        printf("%d¥n", num);
    }
}
```

# 繰り返し③(for)

3個目の繰り返しは、前の二つとは若干異なり、ループする回数を指定出来るという長所を持つ(もちろん、例外もあるけどね☆)

## ○構文

```
for (初期化式;条件式 ;カウンタ) {  
    処理;    //条件を満たす間、繰り返す  
}
```

## ○プログラミング実践

### 1から5の総和を求める

例として、do-whileと同じ処理。比べてみてほしい。

```
#include <stdio.h>
void main(void)
{
    int sum = 0;
    int i;
    for (i = 1; i <= 5; i++) {
        sum += i;
    }
    printf("%d\n", sum);
}
```



## ○演習

5つの要素を持つint型配列を用意し、その全ての要素をその要素番号に初期化して表示するプログラムを作成してください。

```
a[0] = 0; a[1] = 1; a[2] = 2;  
a[3] = 3; a[4] = 4;
```

意味は無いけど・・・

初期化する部分はfor文で。

表示する部分はwhile文で。

## ○演習2

scanfで-1が入力されるまで繰り返し整数を入力をさせ、その前までの数値の平均値を表示するプログラムを作成してください。

ただし、入力される数値は0-100とし、それ以外が入力された場合は”a,ri,e,na,i...”と表示してください。

時間がある人はfor文で挑戦ね？

(そんなに変わらない...はず)

# ○解答

```
#include <stdio.h>
void main(void)
{
    int array_Int[5]; int i;
    for (i = 0; i < 5; i++) {
        array_Int[i] = i;
    }

    i = 0;
    while (i < 5) {
        printf("%d¥n", array_Int[i]);
        i++;
    }
}
```

# ○解答2(1/3)

```
// まず、平均を求めるために何が必要かを考える。  
// 複数回、scanfで入力を待ち、-1が出てきたなら、それまでの合計を足した回数で割った  
// つまり、最低限、{入力を代入するもの・合計・足した回数}の三つを保持しておく変数が必要  
// 物を表示する  
// である。
```

```
#include <stdio.h>  
void main(void)  
{  
    // 入力された数の代入用  
    int input;  
    // 入力された数の合計(一度も足してないから、最初は)  
    int sum = 0;  
    // 入力回数(同じく、最初は)  
    int count = 0;  
  
    // プログラム内容の説明  
    printf("0-100の整数を繰り返し入力してください。¥n-1が入力されると処理を抜け、  
    それまでの平均を表示します¥n");
```

# ○解答2 (2/3)

```
// 永遠ループ(-1が入力されるまで、ループを抜けない)
while (1 == 1) {
    // ユーザに整数を入力してもらう
    scanf("%d", &input);

    // もし、入力された整数が-1だったならば...
    if (input == -1) {
        // ループを抜ける
        break;

    // もし、入力された整数が範囲外ならば...
    } else if (input < 0 || input > 100) {
        // 範囲外であることを表示し、
        printf("a,ri,e,na,i...¥n");
        // ループの最初の処理まで戻る
        continue;
    }

    // ここまで処理が通ったということは、入力された整数が
    // 0-100なので、その数を合計に足し合わせ、かつ足した回数を増やす
    sum += input;
    count++;
}
```

## ○解答2 (3/3)

```
// ↓で平均を表示するわけだが...
// 注意したいのは、一度も入力されなかったとき。
// 0で割るわけにはいかないなので条件分岐で避ける。
if (count == 0) {
    printf("一度も入力されませんでした¥n");
} else {
    // doubleを扱う時は%fを使う。
    // また、(double)についてだが、
    // 整数/整数は小数点以下が無視されるため、sumを強制的に
    // double型に変換することで回避した。
    printf("今までの平均は...%6.2lf¥n", (double)sum / count);
}
}
```

## ○解答2 part2

```
while (1 == 1) {
```

の行を

```
for (;;) {
```

に変更するだけ。

for(初期化式;条件式;カウンタ)について、何も条件式に何も書かないと、終了しないループになる。