# 初心者のための C言語講座

#5: for文, while文

# #5-1）for文

今回はループ処理の話です。

例えば、こんな問題があったとします。

「MIS.W」という文字列を100個表示するようなプログラムを作成せよ。

プログラム中に、printf()を100回も書きたくはないですよねぇ……。そんなときに活躍するのが、for文というやつです。

MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MI(以下略)

▲ こんなのを作りたい

# #5-1）for文

　実際に、これを実現したプログラムを見てみましょう。右のProgram5.1を見てください。

　赤字の部分が、for文を用いてループ処理をしているところです。

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i < 100; i++){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.1　misw.c

# #5-1）for文

さて、詳しく見ていきましょう。

```
for(i = 0; i < 100; i++){
    printf("MIS.W¥n");
}
```

for文はざっくり言うと、「初期条件」「ループ条件」「遷移処理」そして「ループ内容」の4つで構成されています。

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i < 100; i++){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.1　misw.c

# #5-1）for文

```
for(i = 0; i < 100; i++){
    printf("MIS.W¥n");
}
```

①

プログラムが実行されると、main関数の上から順に実行されていきます。for文の部分に到達すると、まず「初期条件」に書かれてあることが行われます。

つまり、変数iに0が入った状態で、for文に突入することになります。

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i < 100; i++){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.1　misw.c

# #5-1）for文

```
for(i = 0; i < 100; i++){
    printf("MIS.W¥n");
}
```

②

そして、次に「ループ条件」が確認されます。この「ループ条件」が真ならばfor文の「ループ内容」が実行され、偽ならばfor文部分が全てスキップされます。

iの値は0なので、i < 100を満たします。したがって、「MIS.W」の出力が行われます。

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i < 100; i++){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.1　misw.c

# #5-1）for文

これは、「i = i+1」の意味
（インクリメントという）

```
for(i = 0; i < 100; i++){
    printf("MIS.W¥n");
}
```

③

プログラムは「ループ内容」の命令を行った後、for文の終わりに到達します。ここで、「遷移処理」が行われます。

つまり、1個目の「MIS.W」を出力後に、変数iの値が1つ増加して、iの値が1になることが分かります。

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i < 100; i++){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.1　misw.c

# #5-1）for文

```
for(i = 0; i < 100; i++){
    printf("MIS.W¥n");
}
```

④

「遷移処理」を行ったら、②に戻ります。つまり、「ループ条件」を確認します。真ならば「ループ内容」を実行し、偽ならばfor文部分をスキップします。

iの値は1なので、まだまだi < 100は満たします。したがって、2個目の「MIS.W」が出力されます。

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i < 100; i++){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.1　misw.c

# #5-1）for文

```
for(i = 0; i < 100; i++){
    printf("MIS.W¥n");
}
```

⑤

この作業を延々と繰り返していると、100回目の「MIS.W」を出力した後の「遷移処理」で、iの値が100になります。

こうなると、「ループ条件」を満たさなくなるので、for文によるループが終わります。

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i < 100; i++){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.1　misw.c

# #5-1）for文

軽く問題を解いてみましょう。以下の(1)～(2)のプログラムは、一体何個の「MIS.W」を出力するでしょうか？

(1)

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i <= 10; i = i +2){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.2　misw2.c

(2)

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 10; i > 0; i--){
        printf("MIS.W¥n");
    }
    return 0;
}
```

「i = i - 1」のこと!!

▲ Program5.3　misw3.c

# #5-1）for文

(1)

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i <= 10; i = i +2){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program4.2　misw2.c

1回目のループ開始時　……　i = 0
2回目のループ開始時　……　i = 2
3回目のループ開始時　……　i = 4
4回目のループ開始時　……　i = 6
5回目のループ開始時　……　i = 8
6回目のループ開始時　……　i = 10
7回目のループ開始時　……　i = 12

ループ条件を満たさないので、7回目のループは実行されません。したがって答えは、

6回

# #5-1）for文

(2)

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 10; i > 0; i--){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.3　misw3.c

1回目のループ開始時　……　i = 10

2回目のループ開始時　……　i = 9

3回目のループ開始時　……　i = 8

・・・・・

9回目のループ開始時　……　i = 2

10回目のループ開始時　……　i = 1

11回目のループ開始時　……　i = 0

ループ条件を満たさないので、11回目のループは実行されません。したがって答えは、

10回

# #5-2）while文

　さて、今度は同じ問題をwhile文を使って解きたいと思います。

　また「MIS.W」を100回出力すると思うとウンザリするかもしれませんが頑張ってください。

MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MIS.W
MI(以下略)

▲ またこんなのを作りたい

# #5-2）while文

while文を使うと、右のProgram5.4のようになります。

赤字の部分が、while文を用いてループ処理をしているところです。

```
#include <stdio.h>

int main(void){
    int i = 0;
    while(i < 100){
        printf("MIS.W¥n");
        i++;
    }
    return 0;
}
```

▲ Program5.4　misw4.c

# #5-2）while文

```
while(i < 100){
    printf("MIS.W¥n");
    i++;
}
```

while文は、「ループ条件」と「ループ内容」の2つの要素しかありません。したがって、「ループ内容」にiを増やす処理も組み込まないといけないのです。

while文は「ループ条件」を満たしている間だけ、「ループ内容」を実行すると覚えましょう。

```c
#include <stdio.h>

int main(void){
    int i = 0;
    while(i < 100){
        printf("MIS.W¥n");
        i++;
    }
    return 0;
}
```

▲ Program5.4　misw4.c

# #5-3) break

さて、次はbreakのお話です。

右のProgram5.5を見てください。このプログラムは、何個の「MIS.W」を出力するでしょう？

```
#include <stdio.h>

int main(void){
    int i = 0;
    while(i < 100){
        if(i == 50) break;
        printf("MIS.W¥n");
        i++;
    }
    return 0;
}
```

▲ Program5.5　misw5.c

# #5-3) break

答えは50個です。

なぜかというと、iが50のときに、以下のような分岐がありますよね。

```
if(i == 50) break;
```

breakとは「今いるループ文から抜けろ」という意味です。したがって、breakされると強制的にwhile文から抜け、そのままreturn 0まで行ってしまうので、50個までしか出力されないということになるのです。

```
#include <stdio.h>

int main(void){
    int i = 0;
    while(i < 100){
        if(i == 50) break;
        printf("MIS.W¥n");
        i++;
    }
    return 0;
}
```

▲ Program5.5　misw5.c

# #5-4) continue

breakの次は、continueです。

またまた右のProgram5.6を見てください。このプログラムは、何個の「MIS.W」を出力するでしょう？

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i < 100; i++){
        if(i == 10) continue;
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.6　misw6.c

# #5-4）continue

答えは99個です。

if(i == 10) continue;

continueにあたると、それ以降のループ文内の処理はスキップされます。つまり、iが10のときのループでは、「MIS.W」の出力がスキップされるのです。

しかし、continueではループ文自体は継続します。だから、iが10のときにcontinue処理をしても、iが11、12、13……のループは普通に行われます。ゆえに「MIS.W」の個数は、1回だけスキップされるだけなので、99個となるのです。

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 0; i < 100; i++){
        if(i == 10) continue;
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.6　misw6.c

# #5-4) continue

continueを使ったプログラムをもう1つ作ってみました。実行結果を見ると、continueが何をやっているのかよく理解できると思います。

```
#include <stdio.h>

int main(void){
    int i;
    for(i = 1; i <= 5; i++){
        printf("A%d¥n", i)
        if(i >= 4) continue;
        printf("B%d¥n", i);
    }
    return 0;
}
```

▲ Program5.7　continue.c

```
A1
B1
A2
B2
A3
B3
A4
A5
```

▲ Result5.7　continue.cの実行結果

# Column）無限ループ

皆さん、「無限ループ」という言葉を1度は聞いたことがあるかと思います。文字通り永遠にループすることなのですが、これもC言語で実装することができます。しかも簡単です。

右のProgram5.8は、無限に「MIS.W」を出力しまくるプログラムです。このプログラムを実行したが最後、宇宙が終わるその日まで、コンピュータの画面には無数の「MIS.W」が表示されていくのです。~~壮大ですね。~~

```
#include <stdio.h>

int main(void){
    int i = 0;
    while(1){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.8　mugen.c

# Column）無限ループ

なぜ、これが無限ループになるのかと言いますと、while文のループ条件が「1」になっているからです。

C言語において、条件式として「1」と書いたときは、それはすなわち「真」の意味になります。逆に条件式として「0」と書いたときは、それはすなわち「偽」の意味になります。

つまり、ここではwhile文の条件式が常に正しいことになるため、永遠にループから抜け出せない状態になっているわけです。

```c
#include <stdio.h>

int main(void){
    int i = 0;
    while(1){
        printf("MIS.W¥n");
    }
    return 0;
}
```

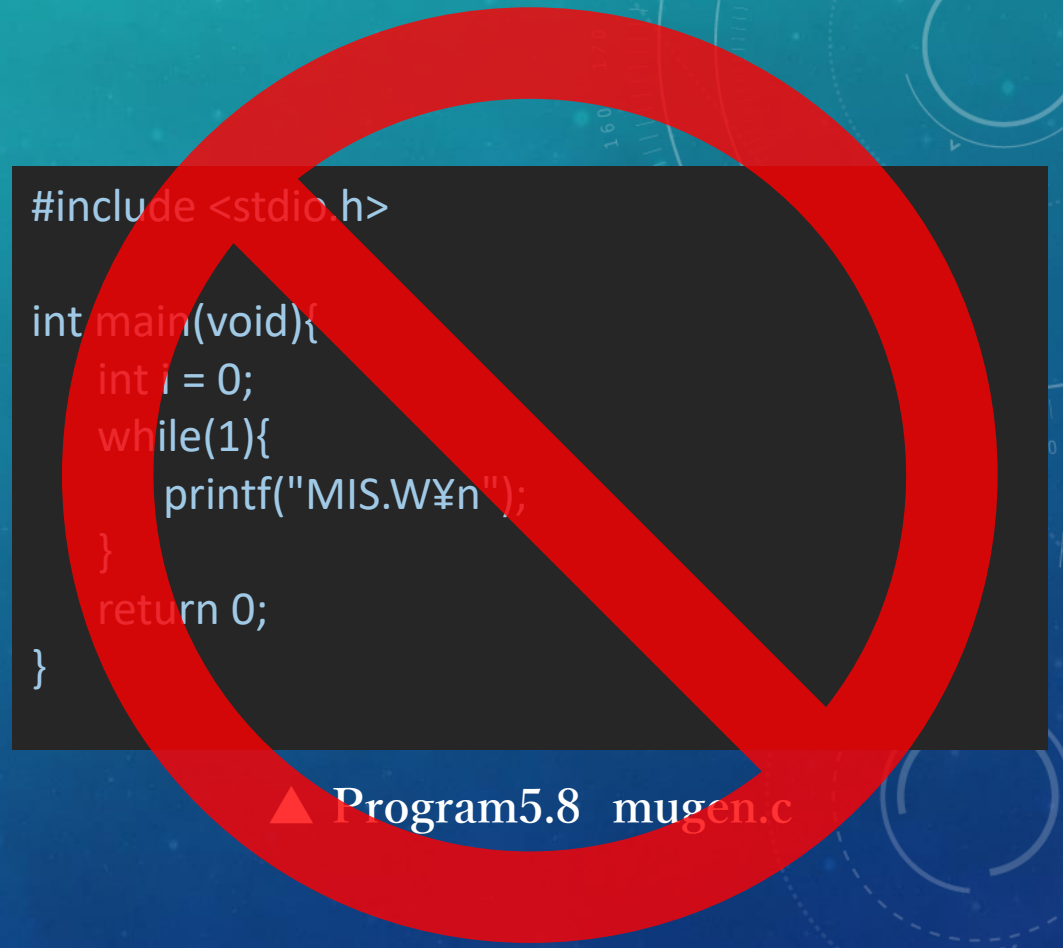
▲ Program5.8　mugen.c

# Column）無限ループ

基本的に、無限ループはそのまま使いません。そのまま使うと、永遠にプログラムが終了しないからです。breakに代表されるような、何かしらのループを抜ける手段と一緒に使うのが普通です。

万が一、無限ループのプログラムを実行してしまったときは、コマンドプロンプト上で「Ctrl」と「C」を同時に押すことで強制的にプログラムを終了させてください。~~それか、宇宙が終わるのを待ちましょう。~~

今回の講義は以上です。今回の演習問題は、ちょっと難しいので頑張ってみてください。



```c
#include <stdio.h>

int main(void){
    int i = 0;
    while(1){
        printf("MIS.W¥n");
    }
    return 0;
}
```

▲ Program5.8　mugen.c

# Question5-1）プログラムを続ける？やめる？

以下のような挙動をするプログラムを作成せよ。ただし、実行結果は下記の【実行例】に準ずるようにせよ。

①

「プログラムを続けますか?」と表示する。

②

数字の0か1を入力する。

1を入力した場合は、①に戻る。

0を入力した場合は、「プログラムを終了します」と表示した上で、プログラムを終了する。

【実行例】

プログラムを続けますか?
1 *[Enter]*
プログラムを続けますか?
1 *[Enter]*
プログラムを続けますか?
1 *[Enter]*
プログラムを続けますか?
0 *[Enter]*
プログラムを終了します。

# Answer5-1）プログラムを続ける？やめる？

```
#include <stdio.h>

int main(void){
        int judge = 1;
        while(judge){
            printf("プログラムを続けますか?¥n");
            scanf("%d", &judge);
        }
        printf("プログラムを終了します。¥n");
        return 0;
}
```

▲ Program5.9　zero_one.c

解答例は、左のProgram5.9の通りです。0が入力されたら、while文の条件式が「偽」となるのでループから抜け出せるようになっています。

# Question5-2）二重ループ

以下のProgram5.10を実行したときの、実行結果を記せ。

```
#include <stdio.h>

int main(void){
    int i, j;
    for(i = 1; i <= 9; i++){
        for(j = 1; j <= 9; j++){
            printf("%d ", i * j);
        }
        printf("¥n");
    }
    return 0;
}
```

▲ Program5.10　kuku.c

# Answer5-2）二重ループ

```
1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

▲ Result5.10　kuku.cの実行結果

　プログラムの名前でネタバレしてる感がありますが、答えは左のResult5.10の通りです。二重ループを使うとこんな感じのこともできるようになります。

　「何でこうなったのか分っかんない！」という人は、そこら辺の先輩を捕まえて問い詰めて見てください。多分答えてくれる（はず）。

　ちなみに、このままだと数字が寄りすぎてて九九の表としては見にくいですよね……。そんな時は、プログラムの変換指定文字を「%3d」に変えてみましょう！各数字に3桁分のスペースが割り当てられるので、スッキリするはずですよ。