ソフトゼミ A　第 3 回

for 文，while 文

今回は for 文、while 文を使った「繰り返し」について学びます。

例えば、「(` ・ω・ ´ ) ｼｬｷｰﾝ」という文を 100 回出力したいと思ったときに、printf(“( ` ・ω・ ´) ｼｬｷｰﾝ”); と 100 回入力するのは、コピー&ペーストするにしても少し面倒です。こんな時に役立つのが、繰り返し処理をする「for 文」と「while 文」です。

◆ for 文

■ for 文の基礎

まずは for 文について説明していきます。for 文を使った繰り返しは基本的に次の様に書かれます。

a03_1.c

```
#include<stdio.h>
int main(void){
        int i;
        for(i=0;i<5;i++){
                printf(“( ` ・ω・ ´ ) ｼｬｷｰﾝ”);
        }
        printf(“¥n”);
        return (0);
}
```

ここで注目してもらいたいのは for 文の後のカッコの中です。式のようなものが 3 つ、セミコロン（ ; ）で区切って書かれていますね。これらは左から「初期化」「制御式」「後処理」といい、for 文はこれらの式を基礎として構成されています。

```
for（i=0 ; i<5 ; i++）
     初期化　制御式　後処理
```

という感じです。

- 初期化 : 繰り返し、つまり for 文の中の処理に入る前に一度だけ実行される処理です。上の例では i に 0 を代入して i の初期値を 0 にしています。
- 制御式 : この条件を満たしている間だけ、繰り返し処理をするという条件式です。上の例では i が 5 未満の間は処理を繰り返し、5 以上になったら終わります。
- 後処理 : for 文の中の処理を一回行い終わるたびに実行される式です。上の例の i++ というのは i=i+1 と同じ意味で、処理を繰り返すたびに i が 1 ずつ増えていきます。

i の様に繰り返し（ループ）の回数を数える変数をループ変数と呼びます。a03_1.c の for 文を言葉で表すと「i を 0 から数えて 5 になるまで繰り返す」となります。i の上限、すなわち制御式の部分を変えてやれば 100 回でも 1000 回でも繰り返すことができます。

また、5 回のループを書くなら for(i=1;i<=5;i++) でもいいんじゃないの？と思う方もいると思います。確かにどちらでも 5 回ループするということに変わりはありません。ただ、のちに「配列」というものを扱うときに、for(i=0;i<5;i++) のように 0 から始める書き方のほうを多用するので、こちらに慣れておくほうが楽です。

- for 文の応用

for 文の基礎で初期化・制御式・後処理について説明しましたが、これらは自由に省略することができます。 for(;i<100;i++) for(i=0; ;i++) for(i=0;i<100; ;) for(i=0; ;) for(; ;) など、文法上全て正しい書き方です。ただしセミコロン（ ; ）は省略不可なので気を付けてください。

これを使うことで無限ループを書くこともできます。

a03_2.c

```
#include<stdio.h>
int main(void){
        int input;
        /*無限ループ*/
        for(;;){
                printf("100 を入力してください¥n");
                scanf("%d", &input);
                if(input==100){break;}
        }
        return (0);
}
```

a04_2.c のプログラムは 100 が入力されるまで無限ループします。このとき、for(; ;)は初期化せず、無条件に繰り返し、後処理もしないという処理をしています。また、for 文内の if 文の中に break と書いてありますね。これはループから強制的に抜け出す命令であり、今回は変数 input＝＝100 という条件が満たされた場合に無限ループを抜けるというようになっています。無限ループを作るときは、必ずループから抜ける条件を書いておくようにしましょう。

for 文を 2 重に書くことで、より高度な処理を行うこともできます。

a03_3.c

```
#include<stdio.h>
int main(void){
	/*ループ変数*/
	int i,j;

	/*3 回　(｀・ω・´) ｼｬｷｰﾝ　と出力するたびに改行*/
	for(i=0;i<10;i++){
		for(j=0;j<3;j++){
			printf("(｀・ω・´) ｼｬｷｰﾝ");
		}
		printf("¥n");
	}
	return (0);
}
```

(｀・ω・´) ｼｬｷｰﾝ を 1 行に 3 回ずつ、10 行書いてみました。変数 i でのループが 1 回繰り返される間に変数 j のループが 3 回繰り返されるようになっています。つまり、i のループの処理が 10 回、j のループの処理が 30 回行われています。

for 文の説明はここで終わりです。お疲れ様でした！

次のページから while 文の説明になります。あと少し、頑張りましょう！

◆ while 文と do－while 文

■ while 文

while 文を使ったプログラムは次のように書きます。出力結果は a03_1.c と同じものになります。

a03_4.c

```
#include<stdio.h>
int main(void){
        /*ループ変数*/
        int i=0;
        while(i<5){
                printf( “( ` ・ ω ・ ´ ) ｼｬｷｰﾝ” );
                i++;
        }
        printf(“¥n”);
        return(0);
}
```

for 文と似た書き方がされていますね。while 文のカッコの中は for 文における制御式のみが書かれます。初期化は while 文の前に、後処理は while 文の内部に書きます。

■ do-while 文

do-while 文を使ったプログラムは次のようになります。これも同じ出力結果になります。

a03_5.c

```
#include<stdio.h>
int main(void){
        /*ループ変数*/
        int i=0;
        do{
                printf( “( ` ・ ω ・ ´ ) ｼｬｷｰﾝ” );
                i++;
        } while(i<5);
        printf(“¥n”);
        return(0);
}
```

do の{}に囲われていますが、**while(i<5)の後にセミコロンがある**ことに注意してください。

また、この do-while 文ではまず内容を実行してからその後、条件分岐が行われることに注意してください。

◆ それぞれの特徴

今回の講習ではいろいろな繰り返しの書き方が出てきました。どのような時にどれを使えばいいの？って思う人もいると思うので、それぞれの特徴をまとめてみました。

- for 文

for 文の良いところは条件、初期化が見やすいところです。基本的には for 文を使うと良いでしょう。

- while 文

while 文の良いところはループから抜ける条件が見やすいところです。脱出条件が複雑な繰り返しや無限ループは while 文で書くと良いでしょう。

- do-while 文

do-while 文は中身が少なくとも一度は実行されます。そういった場合にのみ使うと思って大丈夫です。

大体こんな感じです。深く考えずに、プログラムを書いて慣れてみましょう！

◆ 練習問題

次のプログラムを繰り返しで作り、実行しなさい。

1. (｀・ω・´) ｼｬｷｰﾝ　を 10 行×10 列で出力するプログラム

注) 1 行に 10 回書くとコマンド プロンプトの端まで行ってしまうので、下記のようにして出力結果をテキストファイルにしてみるといいかもしれません。

例) ソースコードが a03_q1.c、出力するファイルにつけたい名前が t.txt の場合、
コマンド プロンプト上で

```
bcc32 a03_q1.c      ←コンパイル
a03_q1 > t.txt      ←実行(出力を t.txt)に変更
```

としてから､テキストエディタで t.txt を開くと 10×10 に並んでいることを確認できます。

2. 7 の倍数が入力されるまでループするプログラム
3. 2 重ループで九九を表示するプログラム