ソフトゼミＡ　第４回 for 文と while 文

■　はじめに

今回は for 文と while 文を学びます。for 文と while 文は『**繰り返し**』のための命令です。＼オワタ／と 100 回出力したいとします。printf(“＼オワタ／”);とキーボードで打った後、99 回コピー&ペーストすればいいわけですが、指が疲れます。こんな処理をするときに『繰り返し』の命令である for 文と while 文が生きてきます。

■　for 文

➢　for 文の基礎

これから for 文について説明していきます。下にサンプルコードがあります。

a04_1.c

```
#include <stdio.h>
int main( void ){
   int i ;
   for( i = 0 ; i < 5 ; i++ ){
      printf( “＼オワタ／” ) ;
   }
   printf(“¥n”);
   return 0 ;
}
```

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

for 文は上の図のような文になっていて、初期化・制御式・後処理の 3 つで構成されています。

✧　初期化…繰り返しが始まる前の一度だけ実行されます。この場合は変数 i を 0 に初期化しています。

✧　制御式…for 文を繰り返すかどうかの条件を書きます。この条件が満たされている間、for 文の中の式が繰り返し実行されます

✧　後処理…for 文の中の式を実行した後の処理を書きます。i++とありますが i = i + 1 と同じ意味です(第 2 回参照)。for 文によって繰り返しが起こるたび i が 1 ずつ増えていきます。

この時、i は繰り返し(ループ)の回数を数えているのでループ変数と呼ばれます。上のプログラムを日本語にすると『i を 0 から数えて 5 になるまで繰り返す』となります。これで 5 回だろうが 100 回だろうが同じ文を書くだけです。

ところで、5 回ループするなら for(i = 1; i <= 5; i++)の方が分かりやすいじゃないか！という方がいらっしゃるかもしれません。どちらでも文法上問題はないのですが、後で「配列」を取り扱う時に for(i = 0; i < 5; i++)の方が色々と楽なのです。詳しくは次回勉強します。

➢　for 文の応用

for 文の初期化・制御式・後処理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 < 10 ;)　for(; ; i++)　for(; ;)　などなど

上の for 文はすべて正しい書き方です。ただしセミコロン(;)は省略してはいけません。

これを使って無限ループを書くことも可能です。

a04_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;
}
```

上のプログラムは 100 が入力されるまで無限ループします。

for 文の括弧内はすべて省略されています、これは初期化せず、無条件に繰り返し、後処理もしないということです。

for 文の中の if 文に break と書いてあります。変数 input が 100、つまりループから抜ける条件を満たした時に実行されます。break は強制的にループから抜ける命令です。無限ループを作るときには必ずループから抜ける条件を書いておきましょう。ちなみに **break はループを抜け**

出すための命令であって、プログラムを終わらせるものではありません。

さらに、for 文を二重に書くことによって高度な処理を行えるようになります。

a04_3.c

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

  /* 5 回＼オワタ／するごとに改行する。  */
  for( i = 0; i < 10; i ++ ){
    for( j = 0; j < 5; j++ ){
      printf( "＼オワタ／" ) ;
    }
    printf( "¥n" ) ;
  }
  return 0;
}
```

＼オワタ／を 50 回横に書くのは無理があるので 1 行に 5 個ずつ、10 行書いてみました。変数が i のループが 1 度繰り返される間に変数が j のループが 5 度繰り返されるようになっています。j のループは中の処理が 50 回、i のループは 10 回行われます。

for 文の説明はここまでです、お疲れ様でした。次は while 文の説明です。あと一息なので頑張りましょう！！

■ while 文と do-while 文

➢ while 文

まず、プログラムを見てみましょう。下のプログラムは a04_1.c と同じ出力になります。

a04_4.c

```
#include <stdio.h>
int main( void ){
  /* ループ変数 */
  int i = 0 ;
  while( i < 5 ){
    printf( "＼オワタ／" ) ;
    i++ ;
  }
  printf( "¥n" ) ;
  return 0 ;
}
```

見ただけで理解できた人もいるのではないでしょうか？

while 文の中には for 文で言うところの制御式だけが書かれています。初期化はなく、必要であれば while 文の前で初期化する必要があります。また、while 文のループの中に後処理である i++ ; が書かれています。これだけです。

➢ do-while 文

プログラムを見てみましょう。これも a04_1.c と同じ出力になります。

a04_5.c

```
#include <stdio.h>
int main( void ){
  /* ループ変数 */
  int i = 0 ;
  do{
    printf( "＼オワタ／" );
    i++ ;
  }while( i < 5 ) ;
  return 0 ;
}
```

do の波括弧に囲われていますが、**while(i＜5)の後にセミコロンがある**のが特徴です。
do-while は do の名前の通り 1 度実行してから条件が参照されます。つまり、最低でも 1 回は条件に関係なく実行されます。

以上で今回の講習の説明を終わります、お疲れ様でした。 ＼オワタ／

■ 練習問題

1. 3 の倍数が入力されるまでループするプログラムを作れ。
2. 二重ループを使って九九を出力するプログラムを作れ 。

■ コラム

今回の講習でたくさんの繰り返し命令が出てきました。どれ使えばいいんじゃ、という人もいるかと思いますので、少し説明していきます。

- for 文の特徴

for 文の良いところは条件、初期化などが見やすいところです。基本的には for 文を使えば間違いありません。ただ、無限ループなどは while 文の方が見やすいと思います。

- while 文の特徴

while 文の良いところはループから抜ける条件が二つ以上の時や条件が複雑な時にそれを簡単に記述できるところです。ただし普通の何回繰り返すというだけの命令では、ややコードが読みづらいです。

- do-while 文の特徴

do-while 文の良いところは少なくとも 1 度は実行される点です。ただ、私は実感したことがありませんので、少なくとも 1 度は実行したいというときに使ってください。