繰り返し処理

同じ手順を繰り返す方法について考える。

画面に文字列リテラル "Hello!\n" を 10 回出力するプログラム

```
main()
{
  printf("Hello!\n");
  printf("Hello!\n");
  printf("Hello!\n");
  printf("Hello!\n");
  printf("Hello!\n");
  printf("Hello!\n");
  printf("Hello!\n");
  printf("Hello!\n");
  printf("Hello!\n");
  printf("Hello!\n");
}
```

という具合にソースプログラムに
10 回書くのも手。しかし、、、

プログラミングがとても面倒。
繰り返し回数が固定されている。

もっと柔軟に繰り返し処理をしたい！

C 言語での繰り返し処理
for 文、while 文、do 文

for 文

繰り返し回数が、文の実行時に決まっている繰り返しを、所定回反復という

所定回反復では何回繰り返したかを数える変数を使う。これを繰り返し変数という。繰り返し変数はたいていの場合整数型

所定回反復には for 文がよく用いられる。反復＝繰り返し（ループ Loop ともいう）

```
for( 式1; 式2; 式3)文
```

式$_1$:繰り返し変数の初期化
式$_2$:繰り返しを続けるかどうかの判定。真なら継続、偽なら for文の終了
式$_3$:繰り返し後の、繰り返し変数の演算
文　:繰り返す文の本体

繰り返し変数はプログラマが用意しなければならない

for 文の具体例

```
int i;

for(i=0; i<10; i++)
  printf("Hello!\n");
```

整数型の変数 i を繰り返し変数として用いる

この例では、1) 繰り返し変数 i をゼロに初期化、2) i の値が 10 未満である限り文(printf)を実行、3) 文の実行後、i の値を 1 増やす。

++ インクリメント演算子　　i++ の実行で i の値が 1 増える

繰り返す処理が複数の文からなる場合はブロックを用いて複文にする(下左)。

```
int i;

for(i=0; i<10; i++){
  printf("Hello!\n");
  printf("%d\n", i);
}
```

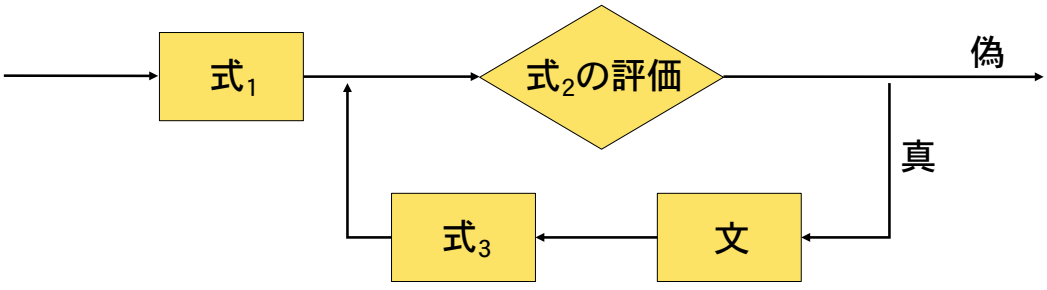
左例と右例では
結果がまったく
異なることに注意!

```
int i;

for(i=0; i<10; i++)
  printf("Hello!\n");
printf("%d\n", i);
```

for 文処理の流れ

for(式$_1$; 式$_2$; 式$_3$)文




```
int i;

for(i=0; i<10; i++)
  printf("Count = %d \n",i);
```

繰り返し変数 i をゼロに初期化。
0 < 10 は真なので printf を実行、その後、
i の値はインクリメントされて 1 となる。
1 < 10 は真なので printf を実行、その後、
i の値はインクリメントされて 2 となる。
...
10 < 10 は偽なので for文の終了。
繰り返しは i の値が 9 になるまで続く。

例 1

```
int i;

for(i=10; i>0; i--)
  printf("Count down %d !\n", i);
```

--:デクリメント演算子

i-- の実行で i の値が1減る

実行結果

```
%./a.out
Count down 10
Count down 9
Count down 8
...
Count down 1
%
```

・繰り返し変数 i を 10 に初期化。
・10 > 0 は真なので printf を実行。その後、i をデクリメント。
・9 > 0 は真なので printf を実行。その後、i をデクリメント。
...
・i の値が 1 になるまで繰り返す。
・i の値が 0 になったとき、0 > 0 は偽なの で、for 文の終了。

注意

繰り返し変数の初期化(式 1)、繰り返しの判定(式 2)、
繰り返し後の演算(式 3)を正しくプログラムしないと、永久に繰り返しが続く
無限ループに陥る場合がある。正しい繰り返しはプログラマの責任。

```
int i;

for(i=0; i<10; i--)
  printf("Count down %d !\n",i);
```

```
%./a.out
Count down 0
Count down -1
Count down -2
...
```

繰り返し後の演算がおかしいので無限ループ。
整数の最小値 INT_MIN をデクリメントすると
INT_MAX になるので、実際はいつかは終了する。

```
int i;

for(i=1; i<10; i++){
  printf("%d\n",i);
  i--;
}
```

```
%./a.out
1
1
1
1
...
```

繰り返し変数を繰り返し文中で操作すると予期せぬ
結果を招くことがあるので、しない。

本当の無限ループ

例 2

次の 2 つの例はいずれも同じ結果を得る。繰り返し変数の初期化の違い。

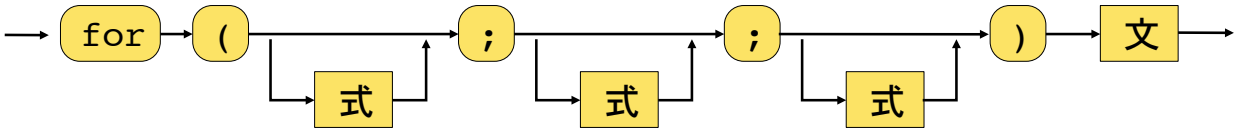
```
int i;

for(i=0; i<10; i++)
  printf("Count = %d \n",i);
```

```
int i;

for(i=1; i<=10; i++)
  printf("Count = %d \n",i-1);
```

for 文の構文図



関連事項

+= 加法代入演算子	i+=2 は i の値を 2 だけ増やす
-= 加法代入演算子	i-=3 は i の値を 3 だけ減らす
*= 乗法代入演算子	i*=2 は i の値を 2 倍する
/= 除法代入演算子	i/=4 は i の値を 4 で割ったもので置き換える

i++ と i+=1 と i=i+1 は同じ意味

i-- と i-=1 と i=i-1 は同じ意味

i*=2 と i=i*2 は同じ意味

i/=2 と i=i/2 は同じ意味

= は代入演算子

100 以下の正の奇数を表示

```
int i;

for(i=1; i<=100; i+=2)
  printf("%d\n", i);
```

100 以下の 2 の二乗を表示

```
int i;

for(i=1; i<=100; i*=2)
  printf("%d\n", i);
```

応用 1

キーボードから整数値を 10 個読み込み、それらの合計を計算するプログラム

```
int sum, data;

sum = 0;  /* 合計を保持する変数を初期化 */

scanf("%d", &data);
sum += data;  /* sum に足していく */

scanf("%d", &data);
sum += data;

scanf("%d", &data);
sum += data;

... /* 同じ手順を10回繰り返す */

printf("合計は %d です\n", sum);
```

scanf を10回用いて整数を10個読み込むプログラム。

プログラムを書くのが大変

読み込む回数を変えるにはソースプログラムを修正してコンパイルし直さないといけない。
（硬直したプログラム）

プログラム例 1

for 文を用いた前例の繰り返し処理

```
int sum, data, count;

sum=0; /* 合計を保持する変数を初期化 */

for(count=0; count<10; count++){
  scanf("%d", &data);
  sum += data;  /* sum に足していく */
}

printf("合計は %d です\n", sum);
```

scanf による入力と加算を複文でひと括りにしてfor文で繰り返す

sum += data; の部分は sum = sum + data; でもよい

入力個数を変えるには for 文の繰り返し条件の部分を変更するだけで良い。
繰り返し条件の 10 を別の変数で置き換えれば、より柔軟なプログラムになる。

応用 2

10 回ではなく、加算する回数も入力して決めるプログラム
（何回繰り返すか、プログラム実行時まで不確定な場合）

```
int sum, data, count, count_MAX;

sum=0; /* 合計を保持する変数を初期化 */

scanf("%d", &count_MAX); /* 加算回数の入力 */

for(count=0; count<count_MAX; count++){
  scanf("%d", &data);
  sum += data; /* sum に足していく */
}

printf("%d 回の合計は %d です\n", count_MAX, sum);
```

count_MAX の値として 0 以下の値が入力された場合、for 文は何も実行しないで終了

応用 3

for 文も文の一つなので、for 文の繰り返し文として for 文を書ける
（for 文の入れ子、多重ループという）

* を縦横 10 x 10 個表示して四角を埋め尽くすプログラム

```
% ./a.out
**********
**********
**********
**********
**********
**********
**********
**********
**********
**********
%
```

printf を用いて * を 10 回続けて出力した後に改行。
これを 10 回繰り返せばよい。（二重ループ）

```
int i, j;

for(i=0; i<10; i++){
  for(j=0; j<10; j++)
    printf("*");
  printf("\n");
}
```

横方向に 10 回 * を出力

改行

上の 2 つの手順を 10 回繰り返す

横方向の繰り返しと縦方向の繰り返しのために繰り返し変数を 2 つ用いる。

問題 1

次の動作をするプログラムを作れ。

```
%./a.out
a b c d e f g h i j k l m n o p q r s t u v w x y z
%
```

この色はプログラムによる出力。

ヒント

for 文を使うこと。文字コードを参照。

文字コード(整数)を 'a' から初めて 1 ずつ加算して表示すれば良い。

表示は、文字コードに対応する文字と空白である。
これを 'z' まで繰り返して最後に改行。

問題 2

下記のような、かけ算の九九の表を表示するプログラムを作れ。

```
% ./a.out
1*1 =  1, 1*2 =  2, 1*3 =  3, ..., 1*9 =  9
2*1 =  2, 2*2 =  4, 2*3 =  6, ..., 2*9 = 18
3*1 =  3, 3*2 =  6, 3*3 =  9, ..., 3*9 = 27
...

9*1 =  9, 9*2 = 18, 9*3 = 27, ..., 9*9 = 81
%
```

この色はプログラムによる出力

ヒント:2つの変数(1 から 9 までの整数)の 2 重ループ。
表示部分は、printf("%2d * %2d = %2d, ", i, j, i*j); として桁をそろえる。
i, j は繰り返し変数。

問題 3

下のような模様を出力するプログラムをつくれ。

```
%./a.out
文字は何にする？#
第一行目の個数は？10
##########
#########
########
#######
######
#####
####
###
##
#
%
```

が 10 個 + 改行
が 9 個 + 改行
が 8 個 + 改行

ヒント:
1）出力する文字を入力
2）繰り返し処理（横方向と縦方向の反復）

一度に 2 重ループを処理しようとしないで、まず最初に横方向の繰り返し処理を考える。この部分を縦方向の繰り返し処理で包んでやる。

この色はプログラムによる出力

問題 4

次の動作をするプログラムを作れ（問題 3 の応用）

```
%./a.out
26 以下の自然数を入力？7
a
ab
abc
abcd
abcde
abcdef
abcdefg
%
```

ヒント:
文字コードを使う

入力した整数値（26以下）を k に格納。

横方向の繰り返し変数を i
縦方向の繰り返し変数を j とする。

'a' + i を文字として繰り返し出力
（ i の値は 0 ~ j ）

この処理を j = 0 ~ k に対して繰り返す。

この色はプログラムによる出力