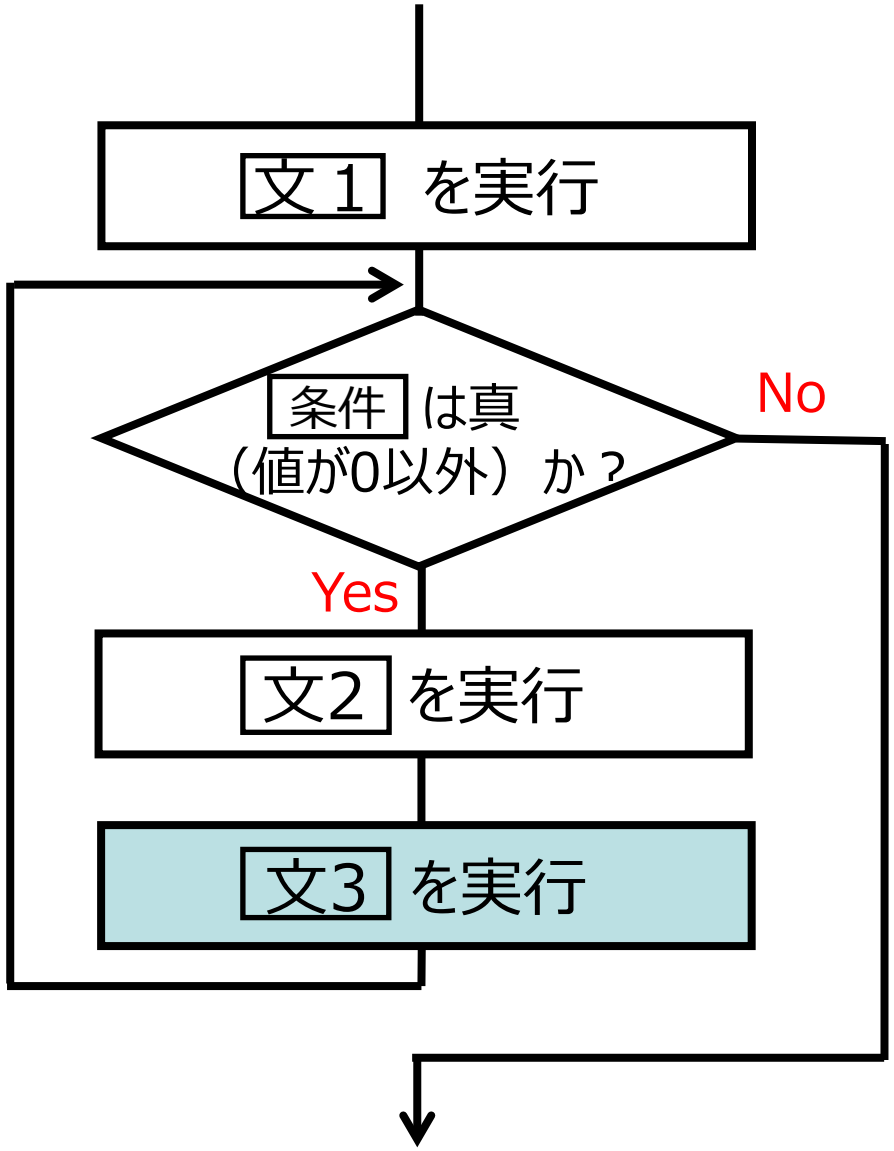
SIT SHIBAURA INSTITUTE OF TECHNOLOGY

情報処理演習II
No. 6

2023. 10. 10

芝浦工業大学　システム理工学部　機械制御システム学科
担当：桑原

for 文



条件の区切りはセミコロン

```
for ( 文1 ; 条件 ; 文3 ) {
        文2 ;
}
```

TABで必ず字下げして読みやすく

文1･文3の主な役割

文1	制御変数の初期設定
文3	制御変数の更新

サンプルプログラム　各自、実行・確認をしてください。

サンプルプログラム 6.1

```
#include   <stdio.h>

void main()
{
        int   i;

        for ( i = 0; i < 10; i++)  {
                printf("i = %d¥n", i);
        }

        printf("Out of the for loop. i = %d¥n", i);
}
```

サンプルプログラム 5.1および5.2 と比較してみてください。

while 文と for 文

使う目的は同じ。使い勝手や可読性などに応じて書き換えられるようにしておくと良いです。

サンプルプログラム 5.1 より抜粋（while 文）

```
int   i = 0;

while ( i < 10 )  {
        printf("i = %d¥n", i);
        i++;
}
```

サンプルプログラム 6.1 より抜粋（for 文）

```
int   i;

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

break 文と continue 文

- break文：

  最も内側の、ループ（for, while, do-while文）やswitch文から脱出（最も内側の1つだけから脱出）。

  次の実行は抜けたループの次の文。

- continue文：

  continue文が実行された位置からループ最後までの文をスキップする。

  次の実行は

  - While、do-while文の場合： 繰返し条件式の判定
  - for文の場合： 制御変数（カウンタ）の更新式「文3」を実行

```
while ( 条件 ) {
    文 ;
}
```

```
for ( 文1 ; 条件 ; 文3 ) {
    文2 ;
}
```

サンプルプログラム　各自、実行・確認をしてください。

サンプルプログラム 6.2　（break文）

```
#include   <stdio.h>

main()
{
    int   i = 0;

    while (1) {
        printf("i = %d¥n", i);
        i++;
        if (i > 10)  break;
    }
    printf("Out of the while loop.¥n");
}
```

while の後の丸括弧内の式の値が「1」なら、

無条件に繰り返しを続ける。

（2であっても0.5であっても同じ結果となる。

「偽」を示す「0」でなければ無限ループを実行）

「もし変数 i の値が 10 より大きければ、break 文を実行」

（つまりwhile ループから抜けて次の実行文のprintfに移る）

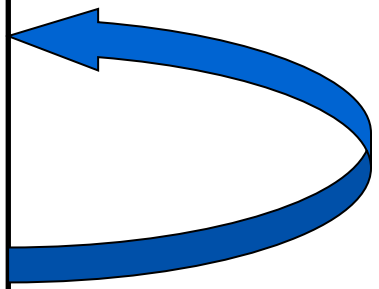
サンプルプログラム　各自、実行・確認をしてください。

サンプルプログラム 6.3（continue文）

```
#include   <stdio.h>

void main()
{
        int   i;

        for (i = 1;  i <= 100;  i++) {
                printf("%d", i);
                if ( i % 10 == 0) {
                        printf("¥n");
                        continue;
                }
                printf("/");
        }
}
```



a % b =　（aをbで割ったときの余り）
よって if 文の条件は、
「i を 10 で割った余りが 0 と等しい」になる
（ただし、これは整数型変数のみで可能）

continue 文実行
これ以降の繰返し処理を中断し、
i++を実行後にループの先頭に戻るため、
「printf(“/”)」は実行されない。

便利な機能（時間情報の取得）

- 関数time： 現在時刻を取得する関数
  - 1970年1月1日0時（GMT）からの経過秒数を整数値で取得
  - 現在を 2013年10月29日朝10時30分 とすると,
    60秒×60分×24時間×{16007日（2013/10/28までの経過日数）}
    ＋ 60秒×60分×｛10.5時間-9時間（GMTとの時差分）｝
    ＝ 1,383,010,200（s）

- 関数timeの利用には，time.hの読込みが必要

  →「#include　<time.h>」をソースファイルの初めに追加

```
#include   <stdio.h>
#include   <time.h>    /* 時刻取得のために追加読込みするヘッダファイル */

void main()
{
     printf("%d¥n", time(0));  /* 1970.01.01からの経過時間を秒数で表示 */
}
```

便利な機能（乱数の発生）

- 乱数の利用 → ゲーム等を作る際に，偶然性を入れる等に利用

  常に変化する量の代表： 時刻 → 乱数の発生に使えないか?

- 乱数の発生手順（例）

Step.1　stdlib.hの読込み

→「#include <stdlib.h>」をソースファイルの初めに追加

（乱数を発生する関数は，ヘッダファイルstdlib内に定義されている）

Step.2. 時間情報を利用した乱数の種を，乱数生成関数にセット

```
srand(time(0));
```

Step.3　乱数発生関数（rand関数）を実行して乱数を生成

```
rand();
```

（注意：↑の括弧の中には何も書かない）

便利な機能（乱数の発生）

サンプルプログラム 6.5 （srand関数，rand関数）

```
#include   <stdio.h>
#include   <time.h>
#include   <stdlib.h>   /* 乱数関連の関数の利用のために読込むヘッダファイル */

void main()
{
     int  x, y;

     /*  時間データを基に乱数の種を生成し、srand関数に渡して乱数生成を準備  */
     srand(time(0));

      /* 「rand()」を実行すると、乱数を1回発生させる。これと剰余演算子により、
          0から9までの数をランダムに発生させ、整数型変数x,yに代入。              */
     x = rand() % 10;
     y = rand() % 10;

     printf("スタート座標です。今、あなたは (x,y)=(%d, %d) にいます。¥n", x, y);
}
```