SIT SHIBAURA INSTITUTE OF TECHNOLOGY

情報処理演習II
No. 5

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

前回のフォロー

１．if文について

```
if (条件A){
   条件Aのときの処理;
} else if (条件B){
   条件Bのときの処理;
} else if (条件C){
   条件Cのときの処理;
} else {
   条件A~Cでないのときの処理;
}
```

前回のフォロー

２．switch文について

```
switch(key){
    case a:
        key==aのときの処理1;
        key==aのときの処理2;
        break;
    case b:
        {
            int x = 0;
            key==bのときの処理;
        }
        break;
    default:
        break;
}
```

case文内に複数行書いてもOK

{}でくくれば、case文内でのみ有効な変数を宣言できる。
（ {} で変数xを使おうとするとコンパイルエラーがでる。）

前回のフォロー

３．演算子の結合順位について

様々な演算が混在する場合、どの演算が優先されるか?

ex.） 乗除、加減、比較（含 NOT）、AND、OR、代入

優先度大 ⟷ 優先度小

→ 読みやすくするために、「（）」を付けておくと良い

ex.） if(((x%2 == 0) && (y != 0)) || (z == 0))

前回のフォロー

４．scanfとprintfの混在

正：scanf(“%lf“, &x);　printf(“%lf“, x)

誤：scanf(“%lf¥n“, &x); ←scanf内で改行しない。

入力してもプログラムに制御が戻らなくなる。

printf(“%lf¥n“, &x); ←printfで&を使うと、変数の値ではなく、

変数のアドレスを参照することになる。

※「コンパイルしていない」と判断される提出物は、採点対象外です。
※他人の提出課題のコピペはやめましょう。
相談はOKですが、必ず自分で書いて確認したものを提出してください。
※課題提出が終わったら退室OKです。

入れ子構造

条件文の中に条件文がある構造
- 困ったらフローチャートを書いてみる。
- 条件分岐するときは、MECEを意識する。

```
if (x >= 100) {          // xが100以上なら
        if (y < 1000) {  // yが1000未満のときは
                z = 10;          // zに10を代入
        } else {                 // そうでなければ
                z = 0;           // zに0を代入
        }                        //
} else {                 // xが100未満なら
        z = 1000;        // zに1000を代入
}
```

- **Mutually（お互いに）**
- **Exclusive（重複せず）**
- **Collectively（全体に）**
- **Exhaustive（漏れがない）**

制御文

- ある条件を判断して何かを実行する

例1：もし明日雨ならば　　傘を持っていく
　　そうでなければ　傘を干す

例2：天気が　晴れならば　帽子を持っていく
　　天気が　雨ならば　　傘を持っていく
　　天気が　曇りならば　　折り畳み傘を持っていく

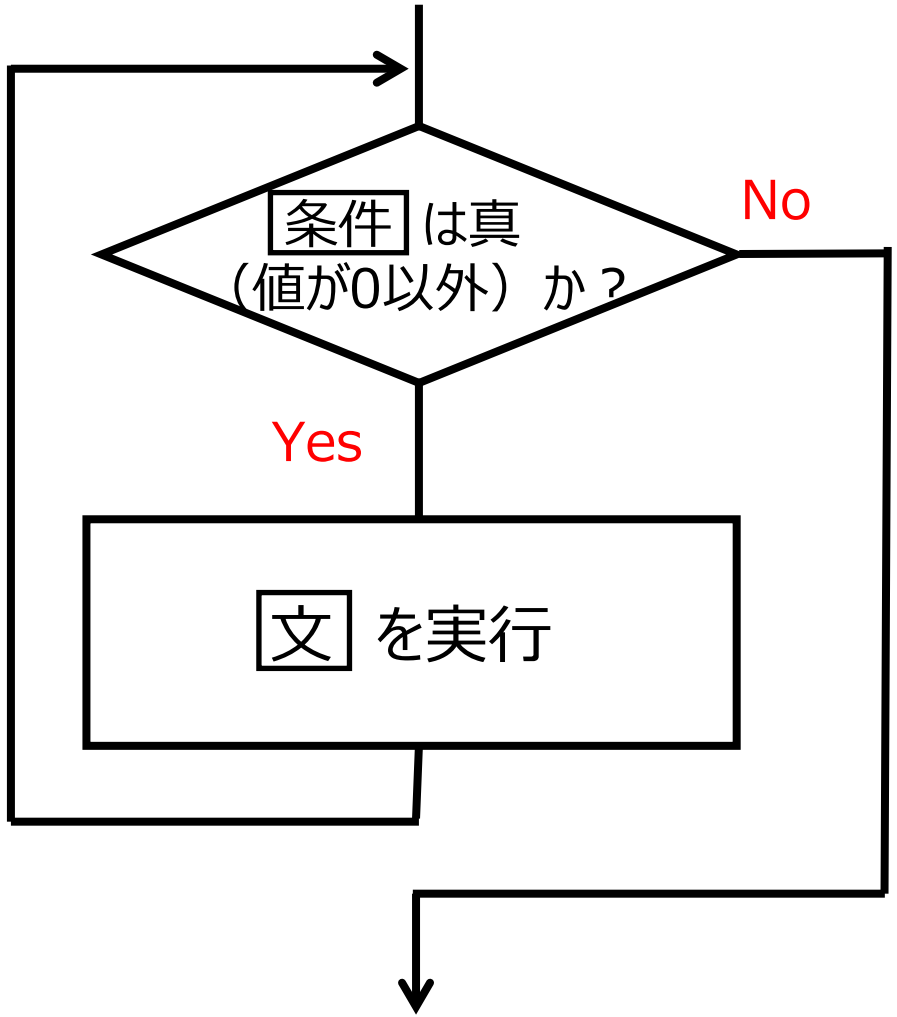
if-else 文
または
switch 文

- 繰り返し何かを実行する

例3：繰り返し 100 円ずつ貯金する
例4：貯金が10000円になるまで
　　繰り返し100円ずつ貯金する

while 文
または
for 文

while 文




```
while ( 条件 ) {
    文 ;
}
```

複数行でもよい.
一行のみの場合は
while (条件) 文 ;
としてもよい.

while 文の例：

```
while ( n == 1 ){
    printf("nは1です. ¥n");
}
```

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 5.1

```
#include   <stdio.h>

void main()
{
        int   i = 0;

        while ( i < 10 ) {
                printf("i = %d¥n", i);
                i = i + 1;
        }

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

「変数 i に（今の i の値）＋1を代入せよ」という処理。

→ **変数 i の値を一つ大きくせよ**という命令（「＝」は「代入演算子」!）。

「i ＝ i ＋ 1」を「i++」または

「++i」と書いてもよい（次スライド参照）

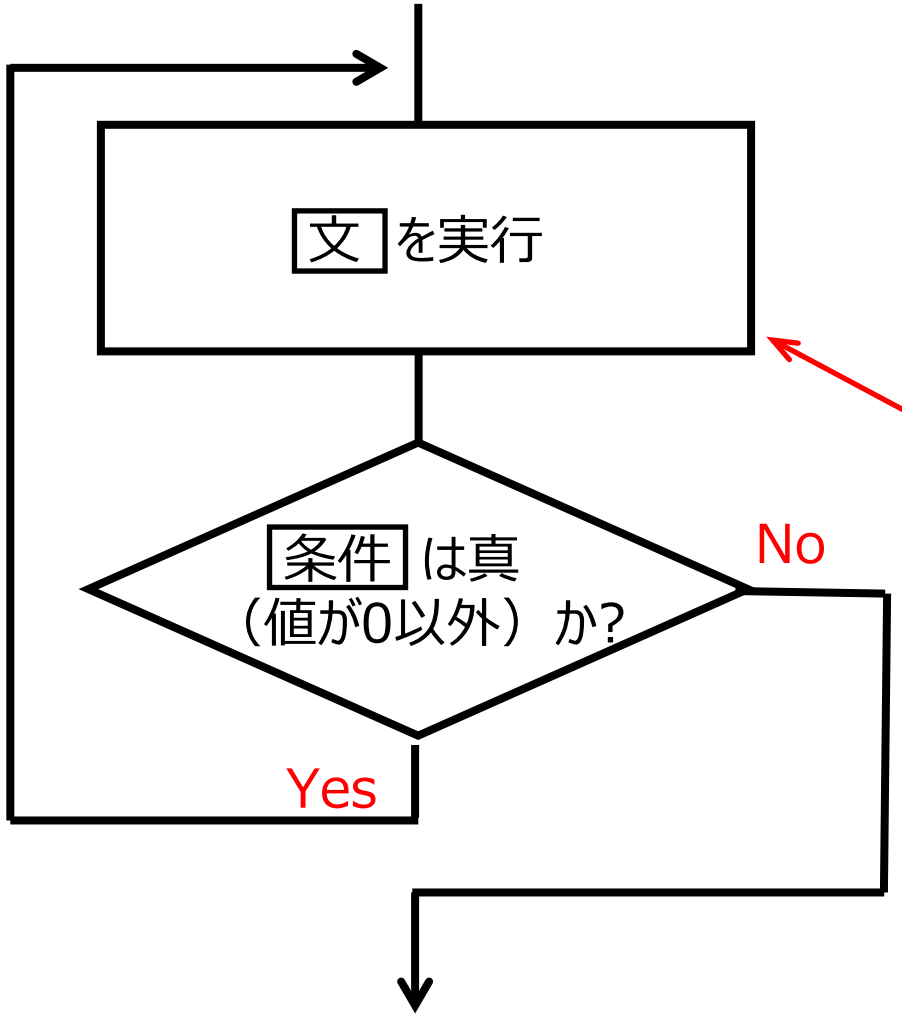
インクリメント/デクリメント演算子

記号	意味	例
++	1 を加える （1大きくする）	++n n++ （変数 n に1を加える）
--	1 を減じる （1減らす）	--n n-- （変数 n から1を減じる）

例

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

do-while 文




```
do {
        文 ;
} while ( 条件 );
```

初回時は判定条件に関係なく
1回だけ実行（while文との相違点）。

注意： do-while文は、do文の後に独立して
while文が付加されたものではない!
whileでの実行文が前に移動したものなので、
while文の後に新たに実行文を書いてはならない。

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 5.2

```
#include   <stdio.h>

void main()
{
        int   i = 0;

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

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

サンプルプログラム 5.1 と
実行結果を比較してみてください。