# 第 4 章　処理の流れを制御しよう

## 概要

条件によって、プログラムの処理の流れを変えたり、処理を繰り返す方法について学びます。
この処理の流れを制御する方法を覚えることによって、高度なプログラムを作成できるようになります。

## この章の目標

比較演算子と論理演算子の使い方をマスターする。
処理の分岐と繰り返し方法について覚える。

---

## 4.1　評価と処理の分岐

---

条件によって処理する内容を変える条件分岐として、if 文と switch 文があります。

---

### ●4.1.1　if 文

4 章　判定しよう－if～else－[pp.40-48]

・書式

```
if( ① ){
    ②;
}
else{
    ③;
}
```

評価が偽の場合の処理③がない場合は、else{}の部分は、省略できます。また、中かっこ{}は、②や③の処理が一つの命令のみの場合は、処理する命令が明らかなので省略できます。

・処理

①式を評価し、真ならば②を処理し、偽ならば③を処理します。
実際の評価では、①式の値を数値として処理し、偽を 0、真を 0 以外の数値として処理します。

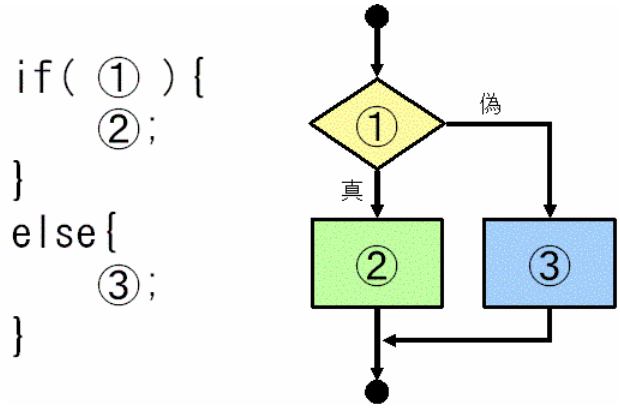


図 4.1-1　if 文の処理の流れ

・条件式
①式の条件式として、次のような比較式がよく用いられます。

```
if( x <= y )
```
x が y 以下か、評価します。

比較式で使用される比較演算子には、関係演算子と等値演算子があります。
また、より複雑な条件式として、次のような論理演算子を用いる表現もあります。

```
if( (x==1) && (y==1) )
```
x が 1 に等しく、かつ、y が 1 に等しいか、評価します。

各演算子の説明は、第 2 章の資料「演算子のまとめ」を参照して下さい。

---

●4.1.2　switch 文

・書式

```
switch( ① )
{
case ②:
    ④;
case ③:
    ⑤;
.
.
.
default:
    ⑥;
}
```

default の部分は、省略できます。

・処理
①の値と等しい case の値（②、③、・・・）の処理（④、⑤、・・・）を実行し、等しい値がない場合は、default の処理⑥を実行します。
なお、次の処理の実行は、各処理（④、⑤、・・・）の最後に、break がある場合とない場合によって異なります。
break がある場合は、switch 文を終わり次の処理に移り、ない場合は、次の case（default）に書かれている処理を実行します。

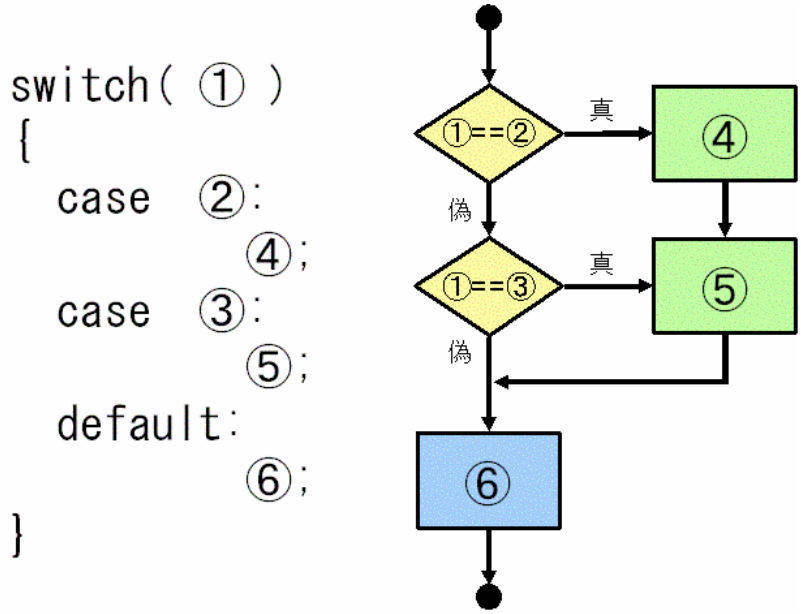


図 4.1-2　switch 文の処理の流れ（break がない場合）

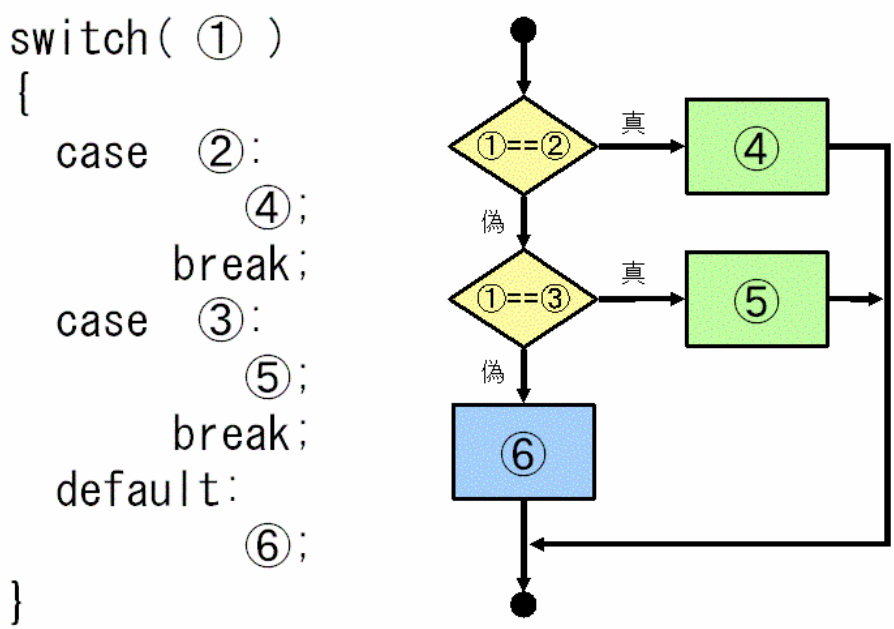


図 4.1-3　switch 文の処理の流れ（break がある場合）

---

●プログラム例
例題 4-1[p. 43]
例題 4-2[p. 44]

（P 例：4.1-1）

```
#include <stdio.h>

main()
{
        int x;

        printf("x = ");
        scanf("%d",&x);
```

```
        if(x >= 5)
                printf("x は、5 以上¥n");
        else
                printf("x は、5 未満¥n");
}
```

・例

```
% x = 3
% x は、5 未満
```

例題 4-3[pp.45-46]

```
 (P 例 : 4.1-2)
#include <stdio.h>

main()
{
        int x;

        printf("x = ");
        scanf("%d",&x);
        switch( x )
        {
                case 1:
                        printf("case 1:¥n");
                        break;
                case 2:
                        printf("case 2:¥n");
                default:
                        printf("default:¥n");
        }
}
```

・例

```
% x = 2
% case 2:
% default:
```

## 4.2　処理の繰り返し

処理を繰り返す方法として、while 文や for 文があります。
5 章　繰り返そう－while ループと for ループ－(pp.54-71)

### ●4.2.1　while 文

・書式

```
while( ① )
{
        ②;
}
```

・処理

①の繰り返し評価を行ってから真ならば②の処理を実行し、再び①の評価を行います。偽ならば while 文から抜け出します。

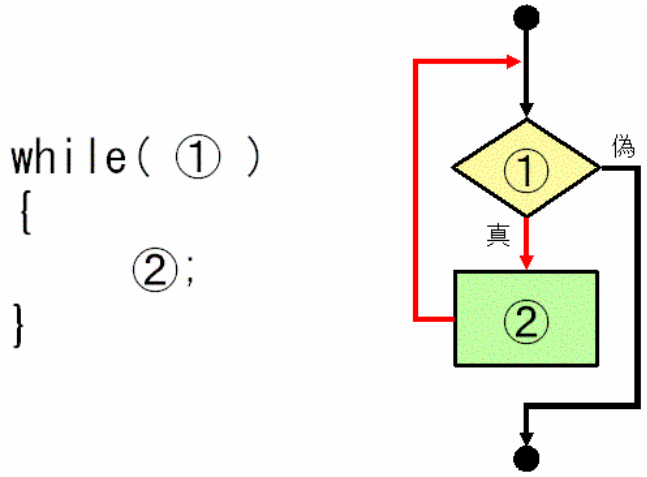


図 4.2-1　while 文の処理の流れ

### ●4.2.2　do-while 文

・書式

```
do{
        ②;
}while( ① );
```

・処理

まず、②の処理を実行してから①の繰り返し評価を行い、真ならば再び②の処理を実行します。偽ならば do-while 文から抜け出します。
従って、①の評価に関係なく、必ず 1 回は②の処理を実行します。

```
do{
        ②;
}while( ① );
```

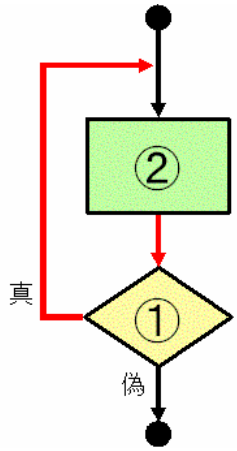


図 4.2-2　do-while 文の処理の流れ

---

●4.2.3　for 文

・書式

```
for (①;②;③) {
        ④;
}
```

・処理

まず、for 文の処理を実行する最初の一回のみ制御変数の初期化（①）を行います。そして、条件式（②）を評価し、真ならば④の処理を実行し、偽ならば for 文から抜け出します。
なお、評価が真で④の処理を実行した後は、制御変数の変更（③）を行い、再び、条件式（②）の評価に戻ります。

```
for (①;②;③) {
        ④;
}
```

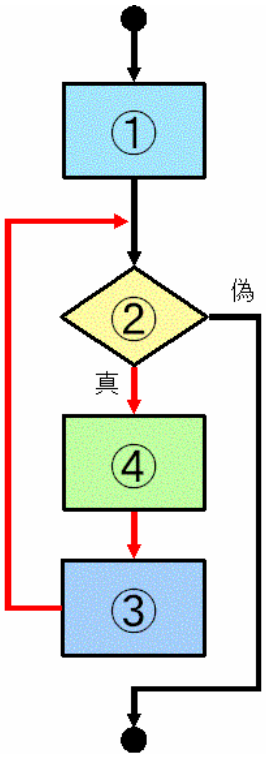


図 4.2-3　for 文の処理の流れ

---

●プログラム例
P 例の（4.2-1）から（4.2-4）までは、1 から 10 までの整数の足し算を行うプログラム例です。

(P 例 : 4.2-1)
```
#include<stdio.h>

main()
{
        int x,y;

        x = y = 0;
        while( x<10 ){
                ++x;
                y += x;
        }
        printf("x = %d, y = %d¥n",x,y);
}
```
・結果
```
% x = 10, y = 55
```

例題 5-3[p.58]

(P 例 : 4.2-2)
```
#include<stdio.h>

main()
{
        int x,y;

        x = y = 0;
        do{
                ++x;
                y += x;
        }while( x<10 );
        printf("x = %d, y = %d¥n",x,y);
}
```
・結果
```
% x = 10, y = 55
```

例題 5-4[p.59]
例題 5-7[p.63]

(P 例 : 4.2-3)
```
#include<stdio.h>

main()
{
        int x,y;

        y = 0;
```

```
        for(x=1;x<=10;x++){
                y += x;
        }
        printf("x = %d, y = %d¥n",x,y);
}
```

・結果

```
% x = 11, y = 55
```

(P 例 : 4.2-4)

```
#include<stdio.h>

main()
{
        int x,y;

        y = 0;
        x = 1;
        for(;;){
                y += x;
                ++x;
                if(x>10) break;
        }
        printf("x = %d, y = %d¥n",x,y);
}
```

・結果

```
% x = 11, y = 55
```

例題 5-8[p.65]

(P 例 : 4.2-5)

```
#include<stdio.h>
main()
{
        int i;

        for(i=0;i<5;i++){
                if(i==3) continue;
                printf("i = %d¥n",i);
        }
}
```

・結果

```
% i = 0
% i = 1
% i = 2
% i = 4
```

例題 5-9[p.66]
例題 5-10[p.67]
2 重ループでの break[p.68]