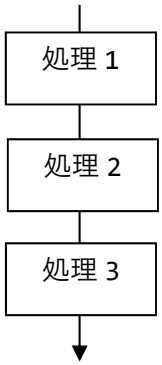
C 言語 008 分岐文

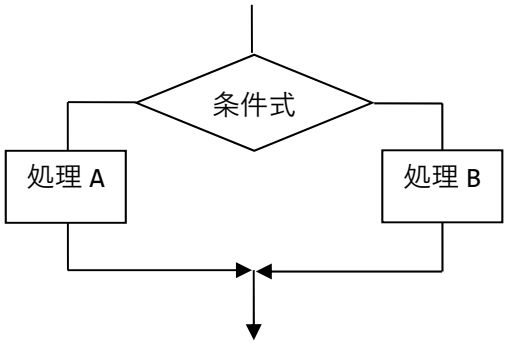
【制御文】

プログラムの流れを制御する文で、「順次」「分岐（選択）」「反復（ループ）」の 3 種類があります。
どのプログラムも、この 3 種類の制御文を組みあせで作られています。

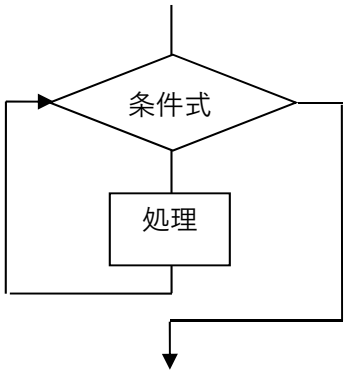
＜順次＞

処理を順番に実行する文です。 ソースコードは書いた順番通り、上から実行されていきます。	

＜分岐・選択＞

条件によって処理が分岐される文です。	

＜反復・ループ＞

条件によって処理を何度も繰り返す文法です。	

【分岐文】

条件によって、処理が分岐される文です。

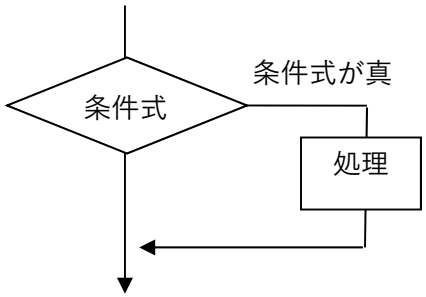
分岐文には「if 文」と「switch 文」があります。

【if 文】

＜if＞

条件が真（該当する）の場合、ブロック内の処理を実行します。

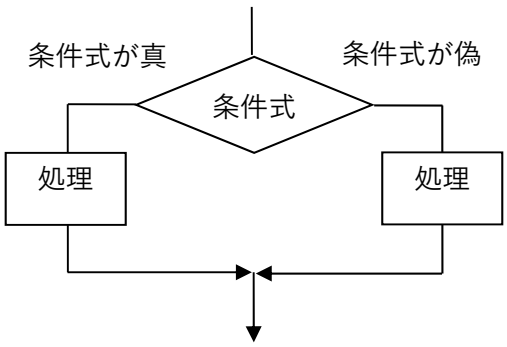
```
if ( 条件式 ) {
    実行する処理
}
```



＜if、else＞

条件が真の場合と偽（該当しない）の場合で、処理が分岐します。

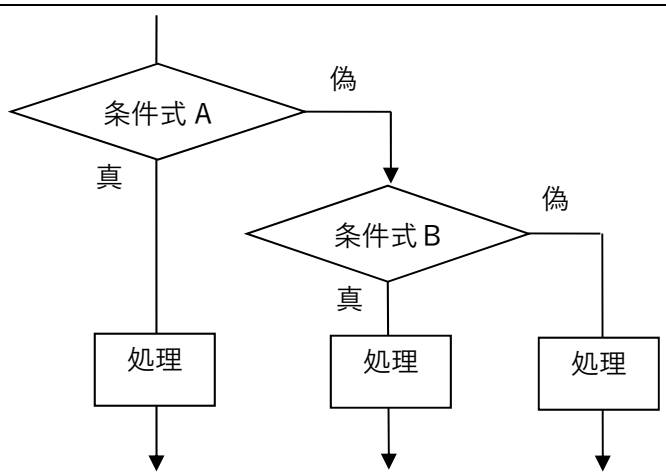
```
if ( 条件式 ) {
    真の場合の処理
}
else {
    偽の場合の処理
}
```



＜if、else if、else＞

複数の条件式を使用して分岐をします。

```
if ( 条件式 A ) {
    条件式 A が真の場合の処理
}
else if ( 条件式 B ) {
    条件式 B が真の場合の処理
}
else {
    全ての条件式が偽の場合の処理
}
```



- 「if と else if」だけの組み合わせも可。
- 「else if」は複数設定できる。
- 記述する順番は必ず「if」「else if」「else」の順。

【条件式】

＜比較演算子（関係演算子）＞

if (var == 10) {	等しい	変数 var が 10 であれば真
if (var != 10) {	等しくない	変数 var が 10 でなければ真
if (var > 10) {	大きい	変数 var が数値 10 より大きければ真
if (var < 10) {	小さい	変数 var が数値 10 より小さければ真
if (var >= 10) {	以上	変数 var が数値 10 以上であれば真
if (var <= 10) {	以下	変数 var が数値 10 以下であれば真

＜論理演算子＞

if (条件式 A & 条件式 B) { if (条件式 A && 条件式 B) {	AND 式 条件式 A と条件式 B の両方が真であれば、式全体は真。
if (条件式 A | 条件式 B) { if (条件式 A || 条件式 B) {	OR 式 条件式 A か条件式 B のどちらかが真であれば、式全体は真。
if (! 条件式) {	NOT 式 条件式が偽であれば真。

＜複雑な条件式＞

条件式に AND 式と OR 式が混在した場合、AND 式が優先されます。

if (条件式 A && 条件式 B || 条件式 C) {

この場合、「条件式 A と B の両方が真」か「条件式 C が真」かの OR 式になります。

if (条件式 A || 条件式 B && 条件式 C) {

この場合、「条件式 A が真」か「条件式 B と C の両方が真」かの OR 式になります。

()で括れば、優先順位を変更できます。

if ((条件式 A || 条件式 B) && 条件式 C) {

この場合、「条件式 A か条件式 B のどちらかが真」と「条件式 C が真」の AND 式になります。

＜例文＞

```
if (var == 10) {
    printf("出力¥n");    /* 変数 var が 10 であれば出力 */
}

if (var == 'a') {
    printf("出力¥n");    /* 変数 var が「a」であれば出力 */
}

if (var >= 5 && var <= 10) {
    printf("出力¥n");    /* 変数 var が 5 以上 10 以下であれば出力される */
}

if ((var1 == 0 && var2 == 1) || (var3 == 0 && var4 == 1)) {
    printf("出力¥n");    /* 「var1=0 で var2=1」か「var3=0 で var4=1」であれば出力 */
}
```

＜「&と&&」「l と ll」＞

```
if ( 条件式 A && 条件式 B ) {
if ( 条件式 A || 条件式 B ) {
```

AND 式では、条件式 A が偽だった場合、式全体が偽になりますので、条件式 B を判定する必要が無くなります。
OR 式では、条件式 A が真だった場合、式全体が真になりますので、条件式 B を判定する必要が無くなります。
必要なければ条件式 B を判定しないのが「&&」「||」になり、判定するのが「&」「|」になります。

```
/* A */
int var = 10;
if (var == 10 || ++var == 11) {
    printf("%d", var);
}
```

```
/* B */
int var = 10;
if (var == 10 | ++var == 11) {
    printf("%d", var);
}
```

A の実行結果は「10」、B の実行結果は「11」になります。
A は最初の条件式で真なため、2 つ目の条件式を見ない、つまり実行しないからです。
B は 2 つ目の条件式も実行しているため「11」にインクリメントされています。

【switch 文】

```
switch (変数名・式) {
    case 値:
        実行する処理
        break;
    default:
        実行する処理
        break;
}
```

- 「switch()」で「変数／式」を指定します。指定した変数の値、指定した式の結果が「case 値:」で指定した値と同じであれば、下記の処理を実行する分岐文です。
- 「case 値:」は複数記述できます。
- 「default:」は、どの case にもあてはまらない場合に下記の処理を実行します。
  「default:」の記述は任意です。使用する場合は最後に記述します。
- 「break;」は switch 文の終了の文です。これの記述を忘れると、switch 文が終了されなくなり、処理が順次実行してしまいます。
- switch 文で指定できる「変数のデータ型」「式の結果のデータ型」は整数（浮動小数点、文字列は不可。文字は整数なので可）でないと使用できません。

＜例文＞

```
int var = 20;
switch (var) {    /* 変数の値は？*/
    case 10:      /* 10 の場合 */
        printf("値は 10¥n");
        break;
    case 20:      /* 20 の場合 */
        printf("値は 20¥n");
        break;
    default:      /* 上記以外の場合 */
        printf("値はそれ以外¥n");
        break;
}
```

実行結果：値は 20

```
int var1 = 10;
int var2 = 10;
switch (var1 + var2) {    /* 式の結果は？*/
    case 10:              /* 10 の場合 */
        printf("結果は 10¥n");
        break;
    case 20:              /* 20 の場合 */
        printf("結果は 20¥n");
        break;
}
```

実行結果：結果は 20

if 文と比べると複雑な条件式は作れませんが、シンプルな分岐に向いています。