3.1 制御用構文とは

通常、C 言語のプログラムは、main 関数の先頭から順に処理を実行する。しかし、**制御用構文**を利用することで、処理を場合分けしたり、特定の処理を繰り返すことができる。

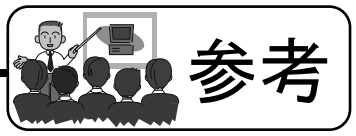
C 言語の制御用構文には、分岐（場合分け）処理に使用する「if 文」と「switch 文」、繰り返し処理に使用する「while 文」、「for 文」、「do～while 文」がある。

分岐文

if 文	条件式が「成立」か「不成立」かによって処理を分岐する（2 分岐）
switch 文	変数や計算式の値を条件に、処理を多数に分岐する（多分岐）

繰り返し（ループ）文

while 文	条件式が「成立」の間、処理を繰り返す ・while 文は条件式のみを記述する ・for 文は条件式のほかに、前処理の式・後処理の式を記述する
for 文	
do～while 文	処理を実行したあと条件式を判定し、「成立」の間繰り返す



～ 構造化プログラミング ～

「**構造化プログラミング**」とは、「順次」「選択構造」「繰り返し」の 3 つの構造のみを利用する手法である。

・順次 …… 上から順番に実行する

・選択 …… 処理を分岐する

・繰り返し …… 処理を繰り返す

これにより、複雑な記述を排除し、プログラムの構造を明確にすることができる。

3.2 if 文

【 例題 1 】入力した 2 つの整数のうち、大きい値を出力する。

【 big.c 】

```
 1 : #include<stdio.h>
 2 : int main(void)
 3 : {
 4 :   int data1, data2;
 5 :
 6 :   printf("input data1 ... ");
 7 :   scanf("%d", &data1);
 8 :
 9 :   printf("input data2 ... ");
10 :   scanf("%d", &data2);
11 :
12 :   /* data1 と data2 の値を比較し、処理を２つに分岐する */
13 :   if( data1 >= data2 ){
14 :     printf("big:%d ¥n", data1); /* data1 >= data2 の場合に実行する */
15 :   }else {
16 :     printf("big:%d ¥n", data2); /* data1 < data2 の場合に実行する */
17 :   }
18 :
19 :   return  0;
20 : }
```

【 実行結果 】

```
C:¥Cpro1>big ↵
input data1 ... 10 ↵
input data2 ... 20 ↵
big:20

C:¥Cpro1>
```

【 プログラム解説 】

7,10 : キーボードから入力した値を変数 data1 および data2 に保存する

13 : 変数 data1 と data2 の値を比較し、処理を 2 つに分岐する

14 : data1 が data2 以上の場合に実行する

16 : data1 より data2 の方が大きい場合に実行する

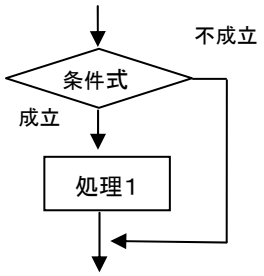
3.2.1 if 文による 2 分岐処理

if 文は、条件式が「成立」か「不成立」かによって、処理を 2 つに分岐する。

if 文の形式

```
if ( 条件式 ) {
    処理 1
}
```

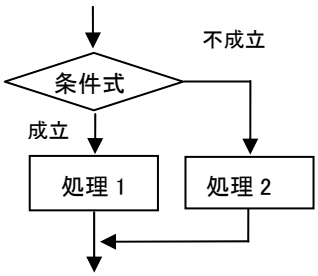
条件式が成立
　　「処理 1」を実行する
条件式が不成立
　　「処理 1」は実行しない



if～else 文の形式

```
if ( 条件式 ) {
    処理 1
} else {
    処理 2
}
```

条件式が成立
　　「処理 1」を実行する
条件式が不成立
　　「処理 2」を実行する



例）

```
if ( data >= 10 ) {
   printf ( " over   10   !! ¥n " ) ;
}
```

← data の値が 10 以上の場合、実行する

例）

```
if ( data >= 10 ) {
      printf ( " over   10   !! ¥n " ) ;
} else {
      pritnf ( " under   10   !! ¥n " ) ;
}
```

← data の値が 10 以上の場合、実行する

← それ以外の場合に実行する

実行する文が 1 文の場合は{　}を省略できるが、分岐する個所を明確化するために{　}を記述した方が良い。

3.2.2　関係演算子と論理演算子

if 文などの条件式で用いる演算子として「**関係演算子**」と「**論理演算子**」がある。

■　関係演算子（比較演算子）

大きい、小さい、等しいなど、値の大小関係を判定するための演算子である。

演算子	意　味
A ＞ B	A が B より大きい
A ＞＝ B	A が B より大きい、もしくは等しい
A ＜ B	A が B より小さい
A ＜＝ B	A が B より小さい、もしくは等しい
A ＝＝ B	A と B が等しい
A ！＝ B	A と B が等しくない

例）

```
int   data = 0 ;

if ( data  <=  0 ) {        ←……  成立 （ 0 <= 0 ）
    printf ( " under   0   ¥n " ) ;
}
if ( data  ==  0 ) {        ←……  成立 （ 0 == 0 ）
    printf ( " equal   ¥n " ) ;
}
if ( data  !=  0 ) {        ←……  不成立（ 0 != 0 ）
    printf ( " not   equal   ¥n " ) ;
}
```

条件式に使う演算子は「==」、代入に使う演算子は「=」である。

「==」と「=」は混同しやすいので、注意する。

■ 論理演算子

いくつかの条件式を組み合わせて判定するための演算子である。

演算子	意　味
式 A　&&　式 B	式 A かつ 式 B （式 A 、式 B が両方成立する場合、全体が「成立」）
式 A　||　式 B	式 A または 式 B （式 A 、式 B の片方、もしくは両方が成立する場合「成立」）
！式 A	式 A の否定 （式 A が不成立の場合「成立」）

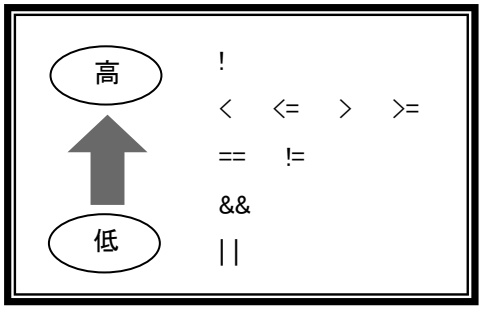
例）

```
float   x = 10.25 ;

if ( x > 0  &&  x < 10 ) {        ←…… 不成立（ 成立 && 不成立 ）
    printf ( " AND   ¥n " ) ;
}
if ( x < 0  ||  x > 10 ) {        ←…… 成立　（ 不成立 || 成立 ）
    printf ( " OR   ¥n " ) ;
}
```

演算子には優先順位がある。優先順位を明確にするために、式を （　）で囲むと良い。

演算子の優先順位（詳細は付録 A 参照）



例）

```
float   x = 10.25 ;
if (  ! ( x == 10 ) ) {        ←…… 成立（ 不成立の否定 ）
    printf ( " NOT   ¥n " ) ;
}
```

3.2.3 if 文による多分岐処理

if～else 文は 1 つの条件式に対して、処理を 2 つに分岐する。この if～else 文を利用し、多分岐処理を記述することもできる。

例）

```
if ( data  >  10 ) {
    printf ( " over   10 !! ¥n " ) ;
} else  if ( data  <  10 ) {
    printf ( " under   10   !! ¥n " ) ;
} else {
    printf ( " Other   ¥n " ) ;
}
```

確認しよう

次の例は、変数 data の値が奇数か偶数かを判定するプログラムの一部である。
空欄にあてはまる語を記述せよ(ヒント:data を 2 で割ったあまりで判定する)。

```
if( ＿＿＿＿＿＿＿＿＿＿ ){
    printf("even   number ¥n");        /＊ 偶数 ＊/
}else{
    printf("odd   number ¥n");         /＊ 奇数 ＊/
}
```

● if 文は処理を 2 つに分岐する制御用構文である。

if 文の形式

```
if ( 条件式 ) {
    処理
}
```

条件式が成立　→「処理」を実行する
条件式が不成立 →「処理」を実行しない

if～else 文の形式

```
if ( 条件式 ) {
    処理 1
} else {
    処理 2
}
```

条件式が成立　→「処理 1」を実行する
条件式が不成立 →「処理 2」を実行する

● 条件式では通常、関係演算子、論理演算子を用いる。

関係演算子(> < >= <= == !=) ← 値の大小関係を比較する

論理演算子(&& || ！) ← いくつかの条件式を組み合わせる