

# 条件判断

C 言語では上から下へと順番に文が実行される。指定した条件に従って実行の流れの分岐を行う場合、**if 文**を用いる。

例) 入力された整数値が正であれば、正と表示するプログラム

```
int main()
{
    int input;

    scanf("%d", &input);
    if( input > 0 ) printf("正の値です!\n");
}
```

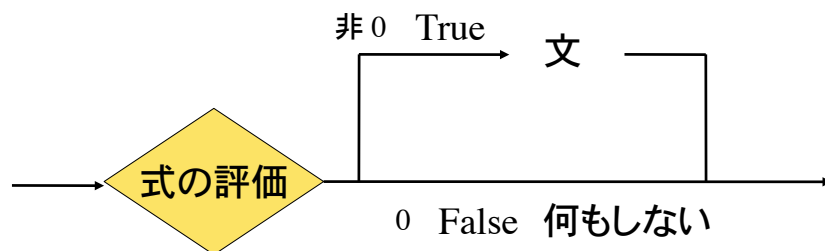
## if 文

### if ( 式 ) 文

if: 英語でもしも～ならば、という意味

式を評価して、その値が非ゼロ(真)であれば文を実行する  
定数や変数名を演算子で結んだものを**式 expression** という。

C では int 型の 0(ゼロ)が**偽 False**、0 以外の値が**真 True**



if 文の流れ(フローチャート)

## 関係演算子 <, >, <=, >=

$a < b$  : a よりも b の値が大きければ 1(真)、そうでなければ 0(偽)

$a > b$  : a よりも b の値が小さければ 1(真)、そうでなければ 0(偽)

$a <= b$  : a の値が b 以下であれば 1(真)、そうでなければ 0(偽)

$a >= b$  : a の値が b 以上であれば 1(真)、そうでなければ 0(偽)

ここで、a と b は int, double 等の型の値を持つ変数

```
int main()
{
    int input;

    scanf("%d", &input);
    if( input > 0) printf("正の値です!\n");
}
           式           文
```

変数 input の値が 0 を越えていれば(0 は含まない)、式は 1(真)であり文 printf() が実行される。0 よりも小さければ式は偽であり、if 文は何もしないで終了。

## 論理演算子

### And 演算子

$a \&\& b$  : a と b が共に真であれば 1(真)そうでなければ 0(偽)

### Or 演算子

$a \|\| b$  : a もしくは b が真であれば 1(真)、そうでなければ 0(偽)

### 等値演算子

$a == b$  : a と b の値が等しければ 1(真)、そうでなければ 0(偽)

### 非等値演算子

$a != b$  : a と b の値が等しくなければ 1(真)、そうでなければ 0(偽)

### 否定演算子

$!a$  : a の値が 0(偽)であれば 1(真)、そうでなければ 0(偽)

!a と  $a==0$  は同じ意味

## よくある間違い

入力された整数値が 7 であれば、ビンゴ!と表示するプログラム

```
int main()
{
    int input;
    printf("整数値を入力:");
    scanf("%d", &input);
    if( input=7 )
        printf("ビンゴ!\n");
}
```

左のプログラムは文法的には正しい。  
しかし、コンパイルは成功するものの、正しく動作しない。

if 文の式が `input=7` となっている。これは変数 `input` に 7 を代入することを意味し、代入式の結果は常に 7 (非 0 なので真) となる。

= は代入演算子。

両辺が等しいことを判定する等値演算子は `==` である。

## 代入演算子再考

代入演算子 = は右辺の式の値を左辺の変数に代入する。

例 `x = 1`

これは代入式である。

代入式自身も値を持つ。その値は代入された値に等しい。

```
int x = 5;
```

```
printf("%d", x);    ← 変数 x の値を表示
printf("%d", x=5); ← 代入式 x=5 の値を表示
```

どちらの表示も 5 となる。

If 分による条件判断、特に等値演算子の使い方に注意

C 言語では、`==` と `=` は全く別の意味を持つ。

## 偶数・奇数の判定

入力した整数値の偶数奇数を判定するプログラム

```
int main()
{
    int input;
    scanf("%d", &input);
    if( input % 2 ) printf("奇数です!\n");
}
```

入力した値 `input` が奇数であれば 2 で割った余りは 1 となる。  
`if` 文の式の評価は 1 (真) であるので、`printf` 文が実行される。  
`input` の値が偶数の時、式の評価は偽となり、何も実行しないで `if` 文は終了。

もちろん次のように書いても結果は同じ

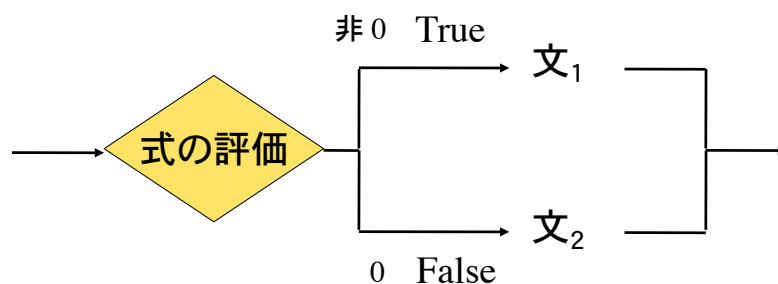
```
if( (input%2)==1 ) printf("奇数です!\n");
```

## もう一つの if 文

```
if( 式 ) 文1 else 文2
```

式が真であれば文<sub>1</sub>を実行し、そうでなければ文<sub>2</sub>を実行する

`else` : 英語でその他の、他の、という意味



## 偶数・奇数の判定その2

```
int main()
{
    int input;

    printf("整数値を入力:");
    scanf("%d", &input);

    if( input % 2 )
        printf("奇数です!\n");
    else
        printf("偶数です!\n");
}
```

if ~ else を使うときにはプログラムを見易くするために改行&段付けをする(タブキーで)

上記の if 文の式を  $input \% 2 == 1$  と書いても同じ結果になる

% は == よりも優先的に評価される。  $(input \% 2) == 1$  としても同じ結果を得る

9

## if 文の構文

プログラム言語の文法上の構造のことを**構文**という

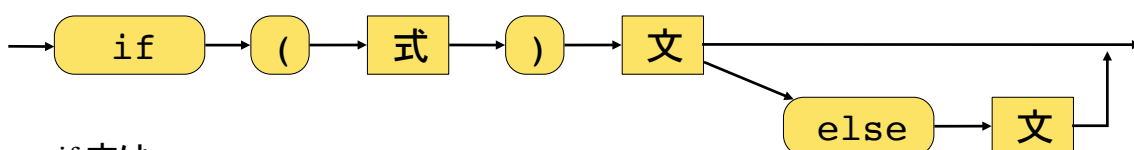
構文を図で表したものを**構文図**という

構文図は、要素と矢印から構成される

要素には、丸囲みで示す**キーワード**と、角囲みで示す**式**や**文**がある

構文図は矢印の方向へ従って進む

if 文の構文図



if 文は、

if ( 式 ) 文 と

if ( 式 ) 文 else 文 の二通りが可能。構文に合わないものは構文エラーになる。

10

# 複文

複数の文を { } で囲んで1つの文にまとめたものを**複文**という

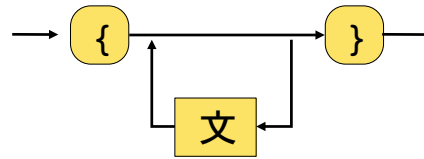
複文は**ブロック** Block ともいう。複文の } の後には ; を付けない!

```
main()
{
    int input;
    scanf("%d", &input);

    if( input % 2 ){
        printf("奇数です!\n");
        printf("あたり!\n");
    }
    else{
        printf("偶数です!\n");
        printf("はずれ!\n");
    }
}
```

この色の部分が複文。2つの文 (printf) を1つにまとめている

複文の構文図



- { }
- { 文 }      すべて複文
- { 文 文 }    一番上は文が無い**空文**
- { 文 文 文 }

....

# if 文の入れ子

if( 式 ) 文

if( 式 ) 文<sub>1</sub> else 文<sub>2</sub>

if 文も文の1つであるので if 文の中に if 文を書くことが出来る  
(**入れ子** Nesting とする)

上の文、文<sub>1</sub>、文<sub>2</sub>として if 文を書くことが出来る

```
int a, b;
...

if (a > b)
    printf("a > b\n");
else
    if(a < b)
        printf("a < b\n");
    else
        printf("a = b\n");
```

if 文は、式が真か偽かの2分岐の条件判断を行う。

論理的には、2分岐を組み合わせることで、複数の条件判断が可能になる。

## 問題 1

日本円とアメリカドルの為替レートを入力して、アメリカドルを日本円に変換するプログラムを作れ。結果の表示は小数点以下 1 桁までとする。

```
% ./a.out
% 今日の為替レートでは、1ドル何円ですか? 105.6
何ドルを両替しますか? 100
100ドルは 10560.0 円です
%
```

円柱の高さと底面の円の半径を入力して、円柱の体積を計算するプログラムを作れ。体積の表示は小数点以下 2 桁までとする。円周率は 3.14159 とする。

```
% ./a.out
% 円柱の高さを入力: 10
底面の円の半径を入力: 2
円柱の体積は 125.66 です
%
```

13

## 問題 2

正の実数を入力し平方根を計算するプログラム。ただし、負の実数を入力した場合は、その旨表示するよう気配りすること。

```
% ./a.out
% 正の実数を入力せよ: 16
16 の平方根は 4.000 です。
%
% ./a.out
% 正の実数を入力せよ: -9
正の実数っていつてるやろ～
%
```

考え方:

- 1) 変数の入力
- 2) 入力した値の正負の判定:  
もし正なら、平方根を計算して表示  
そうでなければ、入力エラー表示

```
if (正)
    平方根の計算と表示
else
    "正の実数じゃないとだめ"の表示
```

14

## 問題 3

整数値を 2 つ入力して割り切れるかどうかを判定するプログラム

割り切れるとは余りがゼロのこと

```
% ./a.out
% 整数を 2 つ入力せよ: 16 5
16 は 5 で割り切れません。
%
% ./a.out
% 整数を 2 つ入力せよ: 24 8
24 は 8 で割り切れます!
%
```

## 問題 4

2次方程式  $x^2 + ax + b = 0$  の解が、実数であるかどうかを判定するプログラム。ただし、係数  $a, b$  は実数として入力する。

```
% ./a.out
% 実数の係数を2つ入力せよ: 4 3
解は実数です。
%
% ./a.out
% 実数の係数を2つ入力せよ: 4 5
解は虚数です。
%
```

判別式  $D = a^2 - 4b$  を使う。



## 問題 5

月(1 から 12)を入力してその季節(春夏秋冬)を出力するプログラム  
不適切な入力はその旨表示して処理すること。

```
% ./a.out
% 月を入力せよ: 5
5 月は春です。
%
% ./a.out
% 月を入力せよ: 12
12 月は冬です。
% ./a.out
% 月を入力せよ: 777
ふざけているのか?
%
```

3, 4, 5 月は春  
6, 7, 8 月は夏      とする  
9, 10, 11 月は秋  
12, 1, 2 月は冬

## 問題 6

月(1 から 12)を入力して、日数を出力するプログラム  
不適切な入力はその旨表示して処理すること。

```
% ./a.out
% 月を入力せよ: 5
5 月は 31 日あります。
%
% ./a.out
% 月を入力せよ: 11
11 月は 30日あります。
% ./a.out
% 月を入力せよ: -9
入力エラーです。
%
```

# シェル (Shell)

Unix システムとユーザの仲立ちをするものとして**シェル** Shell がある。

シェルは、1) ユーザが入力したコマンドの読み込み、2) コマンドの解釈、3) コマンドの実行、を行うインターフェースの役割がある。

シェルによるファイル名の指定

- 1) ファイル名を直に指定: test.c
- 2) **ワイルドカード \* ?** によって指定: test\*.c test?.c
- 3) **限定した範囲内の展開**: test[1-5].c

\* は任意の文字列。従って test\*.c は、  
test.c, test1.c, test2.c, test12.c, testGo.c, testGoGoGo.c 等を含む。

? は任意の 1 文字。従って test?.c は、  
test1.c, test2.c 等を含む。test.c, test12.c, testGo.c 等は含まない。

[1-5] は 1 から 5 までの 1 文字。従って test[1-5].c は、  
test1.c, test2.c, ..., test5.c を表す(展開される)。

# シェル続き

ファイル名の**パターン**(ワイルドカード)

```
% mv *.c gengo1
```

ファイル名が .c で終わる全てのファイルをディレクトリgengo1に移動。

\* は任意の文字列を表す。5-6\*.c は 5-6-1.c, 5-6-2.c などのファイルを含む。

```
% cat *.c
```

拡張子が c であるファイルの内容を表示。

```
% rm *.c
```

拡張子が c であるファイルすべてを削除(絶対にしない)。

```
% ls -l test[1-5]*.c
```

test1a.c, test4GoGo.c などのファイルを一覧表示。