プログラミング演習 第三回 令和4年10月19日

数学関数と条件分岐

明示的な型変換

int 型の値を double 型へ変換をしたい場合がある。

```
int a, b;
double c;

a = 4;
b = 22;
c = b/a;
printf("%lf¥n", c);

c = (double)b/a;

printf("%lf¥n", c);
```

a, b 共に int 型なので、b/a の値は int 5
c は double型変数なので c の値は 5.0

キャスト演算子 ( ) により b の値を double 型へ変換。
22.0/4 の結果は 5.5。c の値は 5.5。

明示的な型変換のことをキャスト cast と言い、キャスト演算子を用いる

(型) 式　　式の値を型としての値に変換

算術関数

倍精度 double 型の実数に対する標準的な算術関数

標準ヘッダファイル `math.h` で定義されている関数

`sin(x), cos(x), tan(x)`：三角関数 与える x の単位はラジアン
`asin(x), acos(x), atan(x)`：逆三角関数

`sqrt(x)`：平方根 負の値 x を与えると実行時エラー

`log(x)`：自然対数 負の値 x を与えると実行時エラー

`exp(x)`：指数関数 e^x

`pow(x, y)`：ベキ乗 x^y

これらの関数を使用するに当たっては `math.h` をインクルードする必要がある

```
#include <math.h>
```

三角関数

sin(x), cos(x), tan(x)等の三角関数はラジアンで角度を与える。度数で角度を与えて三角関数の値を計算するには、度数からラジアンに変換する必要がある。
例えば、30°に対する正弦関数の値を求めるには

$$\sin(\pi/180 \times 30)$$

とする必要がある。π は円周率で、<math.h> で M_PI という名前で定義されている。

```
#include “pch.h”
#include <iostream>
#define _USE_MATH_DEFINES
#include <math.h>
int main(int ac, char *av[])
{
        . . .
     s=sin(M_PI * 30 / 180);
        . . .
}
```

Visual Studioではこの一行を入れる必要がある。

プログラム例

```
#include “pch.h”
#include <iostream>
#include <math.h>

int main(int ac, char * av[])
{
  double x, y;

  printf("実数を入力せよ:");
  scanf_s("%lf", &x);
  y = sin(x);
  printf("sin(%lf) = %lf¥n", x, y);

  return 0;
}
```

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 ac, char *av[])
{
  int input;

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

if 文

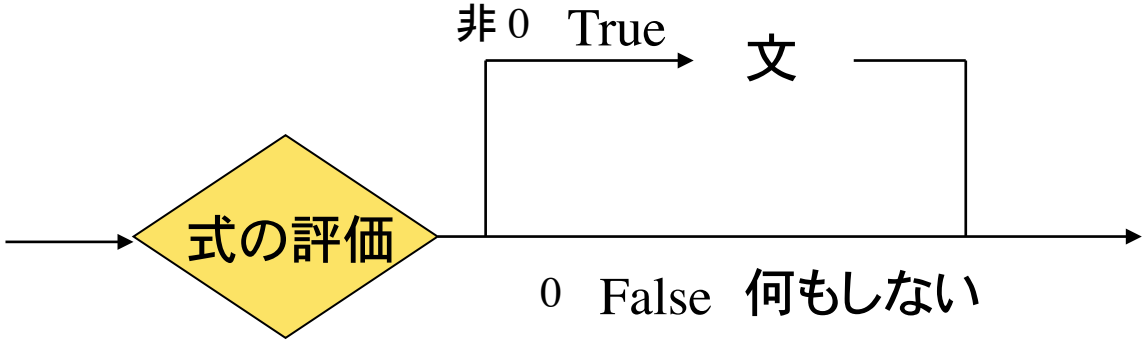
```
if( 式 ) 文
```

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 ac, char *av[])
{
  int input;

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

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

演算子の優先順位

関数	()
キャスト	(型名)
否定、負号	!, -
積、商、余り	*, / , %
和、差	+ , -
関係演算子	>, >=, <, <=
等値、非等値	==, !=
論理積(AND)	&&
論理和(OR)	||
代入	=

高い

順位

低い

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 ac,
           char *av[])
{
  int input;
  printf("整数値を入力:");
  scanf_s("%d", &input);
  if( input=7 )
    printf(“ビンゴ!¥n");
  return 0;
}
```

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);
printf("%d", x=5);
```

← 変数 x の値を表示
← 代入式 x=5 の値を表示

```
double x,y;
int k;
x = k = y = 1.5;
```

x には 1.0、kには 1、y には1.5 が代入される。 代入演算子(=)は右結合で x=k=y=1.5 はx=(k=(y=1.5))と計算される。代入以外の演算子は左結合で、例えば x+y-zは(x+y)ーzと計算される。

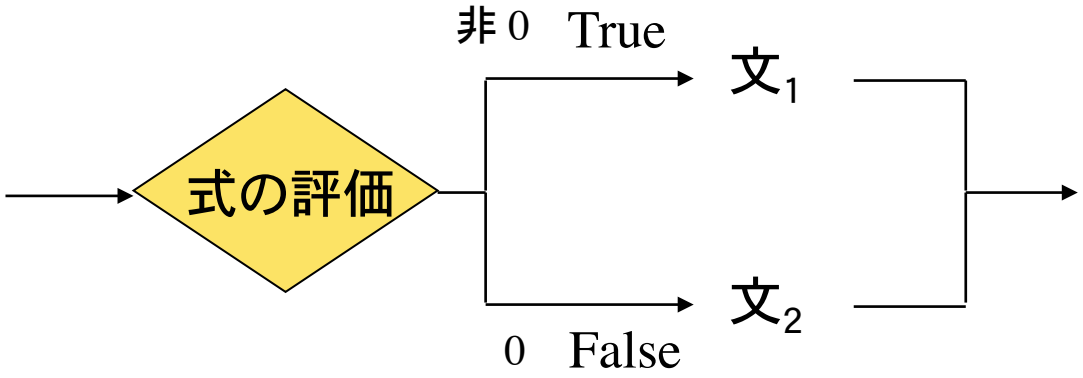
if 文による条件判断、特に等値演算子の使い方に注意
C 言語では、== と = は全く別の意味を持つ。

もう一つの if 文

```
if( 式 ) 文1 else 文2
```

式が真であれば文$_1$を実行し、そうでなければ文$_2$を実行する

else : 英語でその他の、他の、という意味



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

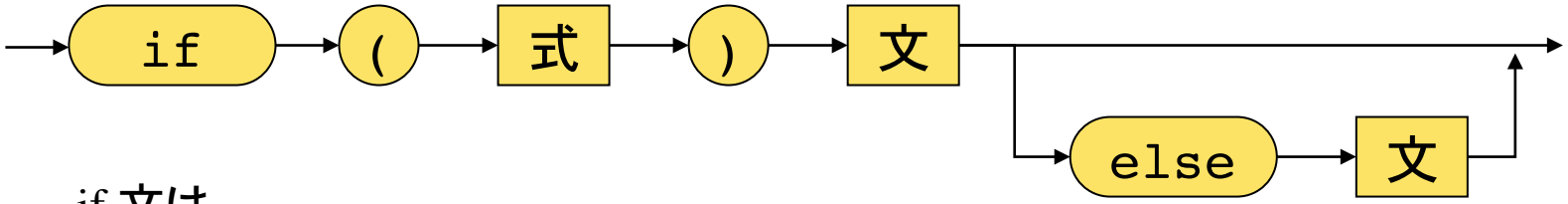
構文を図で表したものを構文図という

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。

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

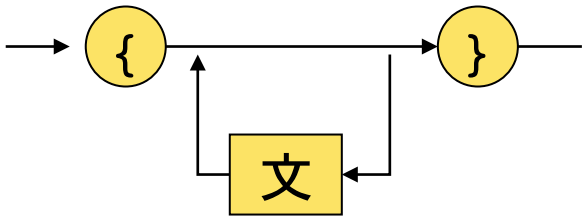
複文はブロック Block ともいう。複文の } の後には ; を付けない!

```
int main(int ac,
             char *av[])
{
  int input;
  scanf_s("%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");
  }
  return 0;
}
```

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 }
{ 文 文 }
{ 文 文 文 }
....

すべて複文
一番上は文が無い空文

if 文の入れ子

```
if( 式 ) 文
```

```
if( 式 ) 文1 else 文2
```

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 と言う）

上の文、文$_1$、文$_2$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により、複数の条件判断が可能になる。

if 文の入れ子(続き)

If文中にif文を入れ子にした場合、意味が曖昧になることがある。

```
if (a >= b)
 if(a == b)
     printf("a == b¥n");
else
     printf("a ? b¥n");
```

この例では、”a ? b”と出力されるのは
どんな場合か?
これをdangling else(中ぶらりのelse)
と呼ぶ。

```
if (a >= b) {
  if(a == b)
     printf("a == b¥n");
} else {
     printf("a < b¥n");
}
```

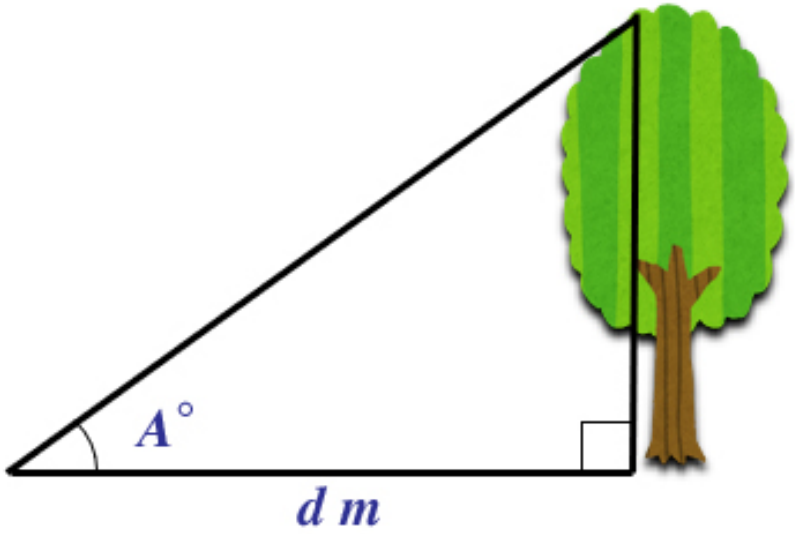
ブロック{…}で囲むことにより else 文に
対応するif文がどれであるかを明確に
する。

問題 1

木の根元から水平にd (m)離れたところから木の先端を見上げたら
水平面とのなす角度（仰角）がA°であった。このとき、木の高さhは

$$h = d\tan(A)$$

で与えられる。距離dと角度A
を入力して木の高さを求める
プログラムを作れ。




```
水平方向の距離を入力せよ: 20
仰角を入力せよ: 40
木の高さ  16.78 mです。
```

問題 2

整数値を二つ入力して、大きい方を出力するプログラムを作れ。
値が同じならば、同じですと出力する。

```
整数を二つ入力せよ: 5 10
大きい方は10です。
```

```
整数を二つ入力せよ: 10 10
二つの数は同じです。
```

問題 3

2次方程式 $ax^2 + bx + c = 0$ の解を求めるプログラムを作成せよ。解が、実数でない場合には、その旨を表示する。係数 a, b, c は実数として入力する。
数値はカンマ(,)で区切って入力するものとする。scanf_sで変換指定の%lfをカンマ(,)で区切る。 scanf_s("%lf,%lf,%lf", &x, &y, &z);

実数の係数を3つ入力せよ: 1, 4, 3
解は–1と–3です。

実数の係数を3つ入力せよ: 1, 4, 5
解は実数ではありません。

判別式 $D = b^2 - 4ac$ を使う。ただし2次係数 a が 0 の時、判定が複雑になるので、a はゼロでないとする。

問題 3（続き）

プログラムの考え方としては、

1. 係数、a,b, cの値を入力する。
2. aの値がゼロであれば、二次方程式でないとしてプログラムを終了する。
3. d = b*b – 4*a*c の値を計算する
4. dが負であれば実数解は持たない。
5. dが0であれば、実数解は一つ x = -b/(2a)
6. dが正（あるいは上の2つのいずれでもない）なら実数解は二つ
   x1 = (-b + √d)／(2a), x2 ＝ (-b - √d)／(2a)

問題 3（続き）

```
if (a == 0) {
  a がゼロなので二次方程式ではない旨のメッセージ
 を出力する
} else {
   dを計算
   if (d < 0) {
     dが負のときの処理を行う;
   } else if(d == 0)
     dがゼロのときの処理を行う;
   } else {
      それ以外（すなわちdが正）の場合の処理を行う;
   }
}
```

ここは
a != 0 の
場合の
処理

注意事項

式を書くときに括弧を忘れないように

```
a = b*c/d*e
```

➔ $a = \frac{b \times c}{d} \times e$

```
a = b*c/(d*e)
   もしくは
a = b*c/d/e
```

➔ $a = \frac{b \times c}{d \times e}$

```
a = (a+b)*(c+d))
```

➔ $a = (b + c)(d + e)$

```
a = 2/3*1.5;
b = 1.5*2/3;
```

順番を変えると結果が違うことに注意。aは1.0 でbは0.0となる。