基礎プログラミング演習 II 教材 (#8)

■ データ型（実数と整数）

いままで数値は整数だけを扱ってきましたが、小数点以下の値（実数）も扱うことができます。ただし C 言語では「型」の概念があり、整数と実数では扱いが異なります。例えば変数にはまず宣言が必要ですが、そこでは変数を「型」と共に宣言します。
（実数には他に double 型もあるが後述。興味のある受講生は教科書 p. 115 以降の 7 章を参照。）

```
int win;
double x,y;
....略...
x=10.0; y=20.0;
while( y < 380.0 ) {
  ....続く
```

前回のサンプルプログラムにおける「`int win;`」の「`int`」は、整数（integer）を扱う int 型の変数 win を宣言するものです。
同じく「`double x, y;`」は実数を扱う double 型の変数 `x` および `y` を宣言しています。

□ 型変換、キャスト

まず定数の表記ですが、`1, 2, 100` などは整数、`1.5` など小数点が付けば実数として扱われます。つまり `2` は整数ですが `2.0` は実数となります。整数と整数の演算（`ex. 1+100`）結果は整数、実数どうし（`10.0+20.5`）は実数となります。問題はその「型」が混在するときです。

int 型変数は整数しか扱えません。そこで int 型変数に実数を代入すると、自動的に型変換が行われ、小数部が切り落とされて整数部だけが代入されます。
つまり `int i; i=123.456;` なら、i には少数以下が切り落とされた 123 が代入されます。

実数変数に整数を代入した場合は、単純に実数化された値が入ります。
`double a; a=123;` なら、123 が実数化されて 123.0 となって a に代入されます。

計算式に整数、実数の値や変数が混在していた場合は、精度の高い方に合わせて型変換が行われてから計算が行われます。`double a; a=1.5 * 3;` なら、1.5 * 3 は 1.5 * 3.0 として計算され、4.5 が a に代入されます。

注意が必要なのは /（割り算）で、整数と整数の割り算は整数となって余りは捨てられますが、実数と実数、または実数と整数の割り算では可能な限りの精度で小数点以下まで求められます。

つまり 10 / 4 は 2 ですが、10.0 / 4.0 は 2.5 です。変数を用いた計算でも同じことで、

```
int i, k; double a, b;
i=10; k=4; a=10.0; b=4.0;
```

の場合、i / k は 2 ですが、a / b は 2.5 です。i / b や a / k のように実数と整数を混在させた場合は、整数側が実数化されて計算され、結果はともに 2.5 になります。

これらは暗黙の型変換と呼ばれますが、明示的に指示して行うこともできます。
もし、`i / k` の計算を実数化して行い、2.5 という結果を実数変数に代入したい場合は、
`double x; x = (double)i / (double)k;` と書きます。
この、値の直前に `()` で囲んで型名を明示指定する方法を「キャスト」と呼びます。

キャストの有効範囲がどこまでか不安になるような場合があり得ます。例えば、
`x = (int) a / k * 100;` のようなケースは、a だけキャストされるのか、全体の計算結果に最後に一度だけキャストされるのか、不安に思うかも知れません。そうしたときは、
`x = (int)(a / k * 100);` のように、カッコをうまく使って記述すると良いでしょう。

□ データの型に合わせた変換文字

右例の printf( ) を見てください。整数は %d 、実数は %f と、変換文字列が使い分けなければなりません。

```
int i;
double d;
i=100;
d=30.0;
printf("int type : %d\n", i);
printf("double type : %f\n", d);
```

実験：右のプログラムを入力して実行し、その結果を表示させて結果を確認してください。
（整数を %f で表示させると？その逆では？）

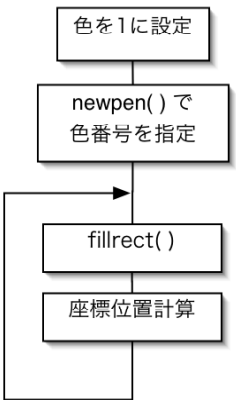
□ 課題 1.

右上のプログラムを修正して、i/30, i/30.0, i*d, i/30*d, i/d*30 がそれぞれどのような結果になるか試してください。
（結果が 0 やおかしな数値になっている場合は型が合っていない可能性があります）

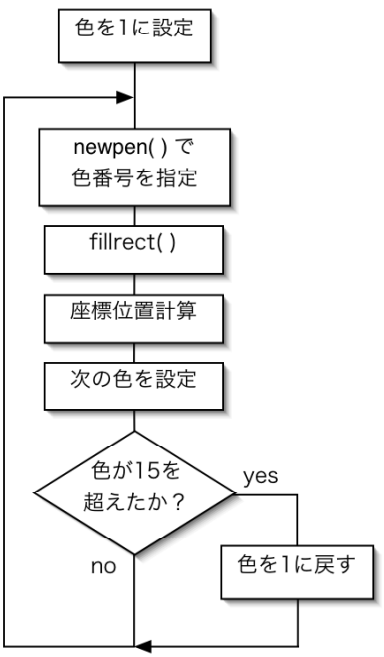
■ 前回の課題 2.（色を変えて棒を描く）における整数と実数の処理

以下のように考えることができます。
・修正内容は色を毎回変えて描くこと
・ newpen() 関数に与えていた色番号を newpen(win, 4); と定数で書かず、変数にすればよい
・ 一本描くたびに色番号の変数の値を変化させれば良い（+1 すれば良いか）
・色が 15 を越えたら 1 に戻す必要あり

元のプログラムの流れ



色を変える場合の流れ



必要な処理は以下のようなものです。

・ 整数変数を一つ宣言
  int color;
・ 最初の値を決める
  color=1;
・ newpen() の色指定を変数に変更
  newpen(win, color);
・ color に毎回 1 ずつ足す
  color=color+1;
・ ただし 15 を越えたら 1 にする
  if(color > 15) color=1;

色のための変数が整数で、座標のための変数が実数である点に注目してください。
このため、color が 15 を越えたかどうかを判定する if 文は整数との比較、y が上限を超えたかどうかを判定する if 文は実数との比較になります。
この型あわせはしなくても同様に動作しますが、常に型を意識し、揃えられるところは揃えて書くよう習慣づけましょう。

■ for を使ったループ

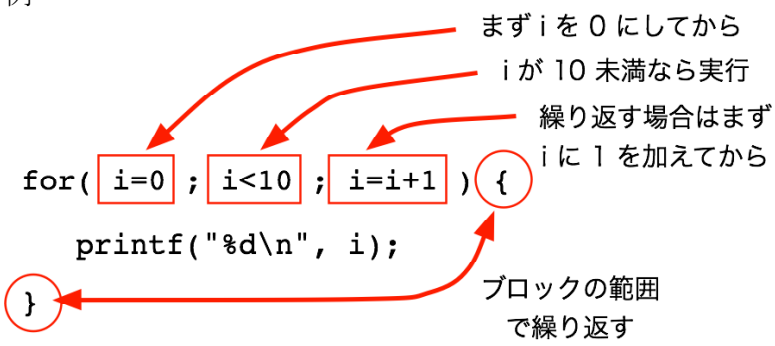
```
main() {
  int i;

  for(i=0; i<10; i=i+1) {
    printf("%d\n", i);
  }
  exit(0);
}
```

右のプログラムを入力して実行してください。0 から 9 までの数字を出力するはずです。

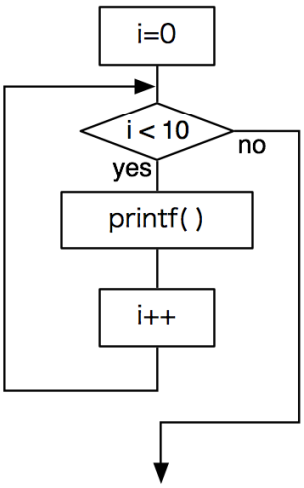
ループを制御しているのが while 文でなく for 文である点に注目してください。

書式：
for(開始時処理 ; 繰り返し条件式 ; 繰り返し毎処理){
　　繰り返す処理
}

例



動作：
for 文は、
・まず開始時処理を実行し、
・その結果が繰り返し条件式に反しておれば何もせず終了
・条件式に当てはまれば続く文またはブロックを実行
・繰り返し毎処理を一度だけ実行してから、
・繰り返し条件判定からくりかえし。
という動作をします。（右図）



プログラムを見ると、for() に続くブロックの処理が 10 回繰り返され、その間に変数 i の値が変化したことがわかると思います。
i が 10 となり、繰り返し条件である `i < 10` を満たさなくなった時点で終了しています。

□ 課題 2.

for を用いて、1 から 50 までの数について、3 あるいは 5 の倍数だけを選んで表示させてください。

□ 課題 3.

for を用いて、1 から 50 までの数について、3 あるいは 5 の倍数がいくつあったか数えて表示してください。

（方法は幾つも考えつくだろう。さまざまトライして最も良いと思われるものを選ぶのが良い。）