コンピュータ基礎実習上級 教材 (#5)

■ for によるループ

□ ループ

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。

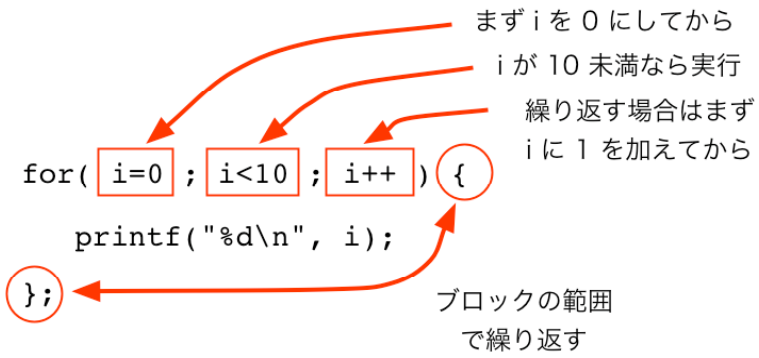
一行しかない printf( ) が 10 回繰り返して実行されています。こうした繰り返し処理のことをループと言います。

```
#include <stdio.h>

/*
  for によるループ 473088 榎田裕一郎
*/

int main() {
  int i;

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

□ for 文

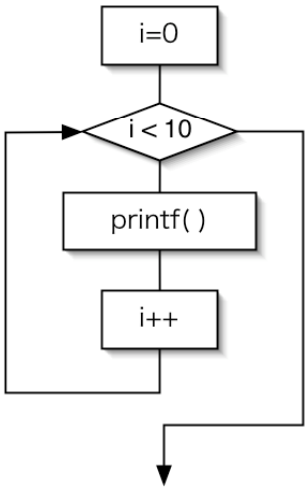
書式：
for(開始時処理 ; 繰り返し条件式 ; 繰り返し毎処理){
　　繰り返す処理
};

例



注意：i++ は「変数 i に 1 を加える」という意味で、「 i = i + 1 」と等価です。（後述）

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あてはまれば何もせず終了
・そうでなければ続くブロック（ { } にくくられた範囲内）を実行
・繰り返し条件判定からくりかえし。
という動作をします。



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。

□ 条件文

i < 10 のように、何かの条件を判定する場合に用いるのが条件文です。(条件文についての詳細は if 文の説明で行います。)

演算子	意味
==	等しい
!=	等しくない
>	左辺が大きい
>=	左辺が等しいか大きい
<	左辺が小さい
<=	左辺が等しいか小さい

==, != を等値演算子、>, < などを関係演算子と呼んでいます。
これらより算術演算子のほうが優先度が高いため、
a < b-1 という記述は a < (b-1) と同じとみなされます。
(左から順に処理されてまず a < b が先に処理されるようにはなりません。)

□ 代入演算子

(例にはでていませんが) a=a+10; を a+=10; と書くこともできます。このように変数への代入処理を対象にした演算子を代入演算子と呼びます。

演算子	利用例 (a を 20、b を 6 とする)	同じ意味の記述
+=	a+=b; (a は 26 となる)	a=a+b;
-=	a-=b; (a は 14 となる)	a=a-b;
*=	a*=b; (a は 120 となる)	a=a*b;
/=	a/=b; (a は 3 となる)	a=a/b;
%=	a%=b; (a は 2 となる)	a=a%b;

□ インクリメント、デクリメント

i++ は「変数 i に 1 を加える」という意味で、「 i = i + 1 」「 i += 1 」と等価です。これをインクリメントと呼びます。同様に減算(デクリメント)もあり、こちらは i--と書きます。

演算子	利用例 (a を 20 とする)	同じ意味の記述
++	a++; (a は 21 となる)	a=a+1; や a+=1;
--	a--; (a は 19 となる)	a=a-1; や a-=1;

これらは ++a 、--a のように使うこともできます。++a は演算に先立って加算され、a++ は演算の後で加算されます。違いの分かる使用例を以下に示します。

利用例 (a を 20 とする)	実行後の結果
b = 6 * a++;	a は 21、b は 120
b = 6 * ++a;	a は 21、b は 126

「 6 * a++ 」は 6 と掛け合わされたあとで加算され、「 6 * ++a 」は 6 と掛け合わされる前に加算されています。慣れないうちは a++; のように、一行だけの記述で、つまり a=a+1; の代わりに使うのがよいでしょう。

■ グラフィクスプログラミング

グラフィクスを描くプログラムを作成しましょう。右のサンプルプログラム loop.c を実行してください。（タイプ入力しなくても講師の Web に教材として置いてあります。）

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "GrWin.h"

int main()
{
  GWopen(0);
  GWindow(0.0, 0.0, 200.0, 200.0);

  GWsrect(10.0, 20.0, 100.0, 30.0, 2);
  GWsrect(10.0, 40.0, 100.0, 50.0, 3);
  GWsrect(10.0, 60.0, 100.0, 70.0, 4);
  GWsrect(10.0, 80.0, 100.0, 90.0, 5);

  return 0;
}
```

□ プログラムの解説

前置きに幾つかの点をまる覚えして下さい。

・ 幾つかある include 文は定型とおもってこのまま書いて下さい。
・ 特に GrWin.h がグラフィクスプログラミングをするときの材料を準備してくれます。
・ GWopen(), GWindwo() も定型だと思ってこのまま書いて下さい。GWopen() は初期化処理、GWindow() は表示する描画ウィンドウのサイズを決めますが、いまは呼び出し方を注意する必要はないでしょう。

最も注目すべきところは GWsrect( ) です。

```
GWsrect(10.0, 20.0, 100.0, 30.0, 2);
```

と書くことで、

座標位置 (10, 20) と (100,30) を端点とする長方形を色番号 2 で描きなさい。

という指示を与えたことになります。
原点(0, 0) は左下です。色番号は以下の通り。

0 : 黒	1 : 栗色	2 : 暗い緑	3 : オリーブ	4 : 濃紺
5 : 紫	6 : 緑青	7 : 灰色	8 : 明るい緑	9 : 薄い水色
10 : 薄い灰色	11 : 青灰色	12 : 濃い灰色	13 : 赤	14 : 緑
15 : 黄	16 : 青	17 : 赤紫	18 : 水色	19 : 白

□ 関数と引数

GWsrect( ) はつまり「塗りつぶした四角形を描く」という機能をもったものでした。
「画面に文字を表示する」機能を持った printf( ) もそうでしたが、C 言語でこうしたひとまとまりの機能を実行するもののことを「関数」と呼んでいます。
GWsrect( ) や GWopen( ) などはすべて関数です。
それぞれのカッコ（ ）の中に与えるパラメタのことを「引数」と呼んでいます。引数は順番と型をきちんと守らなければならないことに注意して下さい。

● 課題 1.

プログラムを右のように修正して、変数を利用して次の四角形の座標位置を計算しながら描画するようにしてください。

```
x=10.0; y=20.0;
GWsrect(x, y, x+100.0, y+10.0, 2);
y=y+20.0;
GWsrect(x, y, x+100.0, y+10.0, 3);
y=y+20.0;
GWsrect(x, y, x+100.0, y+10.0, 4);
y=y+20.0;
GWsrect(x, y, x+100.0, y+10.0, 5);
```

ポイント：
実数型変数 x, y を宣言することを忘れずに。

● 課題 2.

更にプログラムを修正して、ループを使って描画するようにしてください。

考え方のヒント：
・ GWsrect( ) と y=y+20 という全く同じ（またはほぼ同じ）処理が四回繰り返されている。
・ つまりそこは一度だけ書くことにして、それを四回ループさせれば良いのでは？