基礎プログラミング演習 II 教材 (#7)

■ while 落ち穂拾い

```
int count;
count=0;

while( 1 ) {
  printf("%d\n", count);
  count=count+1;
}
```

□ while(1) の 1 は何か

教科書 p.42 中央（最下パラグラフ）「もっと単純な..」以降参照。
・条件として記述するのは論理式である
・式の計算結果は論理値（真・偽）となる。
・C 言語では内部的にこれを 1 と 0 で表現する。（厳密には偽が 0 で、それ以外が真）
・つまり条件「1」は常に真を意味する

□ 文と複文

右に示すように if 文の条件指定の後には処理を一つだけ、あるいは { } でくくって複数の処理を書くことができます。while も同様です。
教科書 p.31 二行目参照（式の最後にセミコロンをつけて「文」とする。代入文は代入式を文にしたもの。）

```
if( 条件 ) 処理...;
if( 条件 ) {
  処理1... ;
  処理2... ;
}
```

```
while( 条件 ) 処理...;
while( 条件 ) {
  処理1... ;
  処理2... ;
}
```

C 言語では { } で一つ以上の文をまとめて一つの文と同じように扱わせることができ、これを複文と呼びます。「ブロック」とも呼びます。
右の二例が共に正しい文法であることが分かりますか。

■ 復習（前回の課題の処理手順を反芻する）

□ 課題 4. : 3 あるいは 5 の倍数を表示（最大 50）

□ 課題 5. : 1 から 10 までの合計を求める

・はじめに合計をゼロとする
・1 回目のループの際には合計に 1 を加え、二回目のループの際には合計に 2 を加え･･･
・ループを終わった時の合計が結果として使える
・まず 10 回ループするプログラムを作り、そこに上記の処理手順を組み込もう

最初に合計をゼロにする必要性に注意。

■ return （2 ページ後の説明：スペースがないのでここに書いています）

（2 ページ後の）サンプルは実行終了を exit( ) ではなく return で行っています。「現在処理している関数（main）の実行を終了して呼び出しもとに戻る」機能があります。この場合は「プログラムを実行した」ところに戻るので、実際にはプログラムそのものが終了します。
なお return は関数ではないので return(0) ではなく return 0 と書きます。
（興味のある受講生は教科書 p.13 の「exit() 関数」を参照。exit() と return の相違の説明あり。）

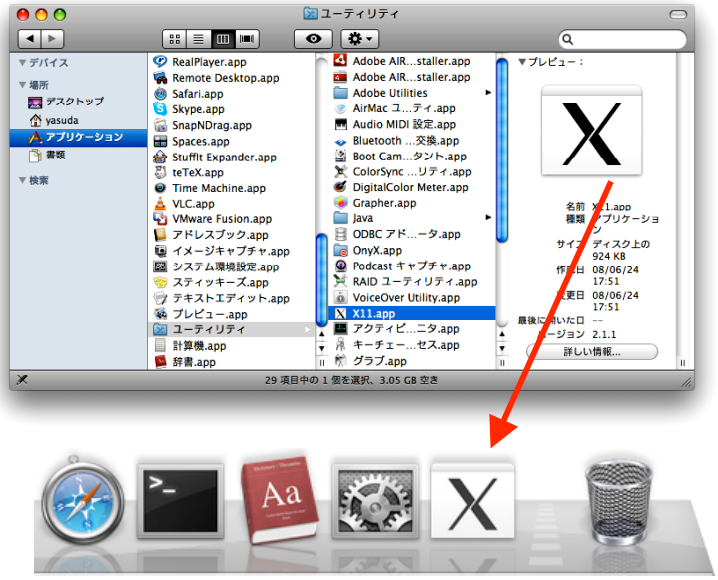
また、このとき冒頭の main() は int main() と書きます。この int と return の関係は後半の「関数」のところで説明します。（興味のある受講生は教科書 p.232 を参照。）

■ グラフィクス

これ以降、EGGX を利用したグラフィクス・プログラミング（画像を描くプログラムを書く）をとりあげます。

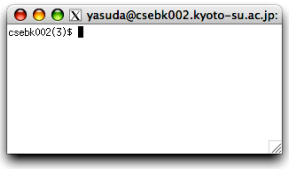
□ 準備（X11 の起動）

EGGX はグラフィクスの表示に X window と呼ばれる仕組みを利用します。X は UNIX システムでは一般的なグラフィクスシステムですが、MacOSX ではオプションとなっているので、EGGX を利用したグラフィクス表示をする時には X を手作業で起動しておく必要があります。右図のように Finder のウィンドウを開き、「アプリケーション」フォルダの中の「ユーティリティ」フォルダにある X11.app アイコン  をダブルクリックして X を起動してください。



X11.app アイコンを Dock にドラッグ&ドロップして登録しておけば、それ以後は Dock のアイコンをクリックするだけで X11 が起動します。

起動すると Terminal ウィンドウが開きます。ターミナルとしてこれを使うこともできますが、混乱するようならクローズボタン  をクリックして閉じてしまえば良いでしょう。



□ egg コマンド

グラフィクスプログラムをコンパイルする場合は、cc ではなく egg コマンドを使います。例えば egsample.c プログラムをコンパイルして実行する場合は以下のようにします。

（egg コマンドによるコンパイルでも –o オプションによる実行ファイル名の指定が可能です。付けないばあいは a.out の名前で作成されます。）

```
$ egg –o egsample egsample.c
gcc -O2 -Wall -oegsample egsample.c -I/usr/bin -L/usr/bin -I/usr/X11R6/include
 -L/usr/X11R6/lib -leggx -lX11 -lm
$ ls
egsample  egsample.c      (←他にもいろいろ表示されるかもしれませんが)
$ ./egsample
```

□ マニュアル

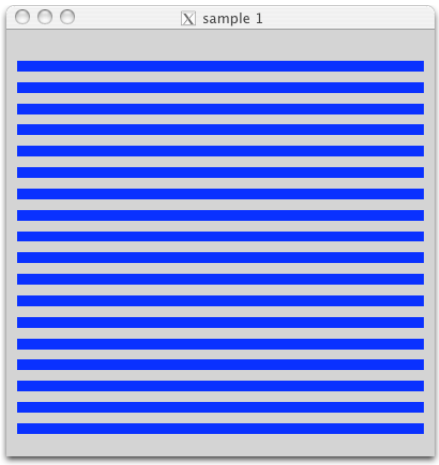
C プログラミングガイド EGGX の章を参照

http://ylb.jp/C/

□ 実験

以下のプログラムを入手して実行してください。
画面上に絵が表示され、何かキーを押すと終了することがわかるでしょう。

```
#include <stdio.h>
#include <stdlib.h>
#include <eggx.h>

int main() {
  int win;
  float x,y;

  win=gopen(400,400);  /* 描画ウィンドウを開く */
  winname(win, "sample 1"); /* 名前をつける */

  newpen(win, 4);  /* 描画する色を設定 */

  x=10.0; y=20.0; /* x,y の初期座標位置 */

  while( y < 380.0 ) {
    fillrect(win, x, y, 380.0, 10.0);
    y=y+20.0;
  }

  ggetch(win); /* キー入力を待つ */
  gclose(win); /* 描画ウィンドウを閉じる */

  return 0;
}
```

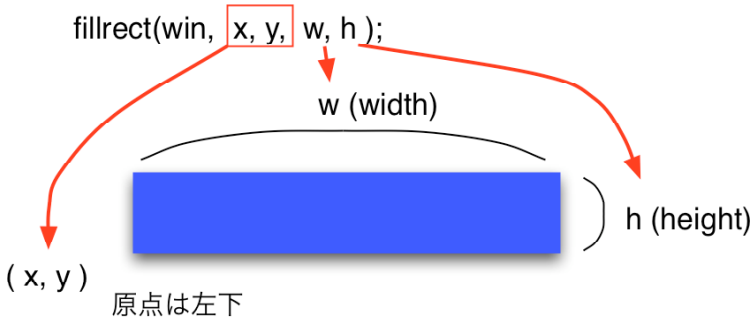


gopen( ) や fillrect( ) などのグラフィクスを描画するために使っているものは、printf( ) などと同じく「関数 (function) 」と呼ばれています。

押さえて欲しいポイント：

・#include <eggx.h> で EGGX に関する準備をする。
・gopen (400, 400) で 400 x 400 画素の描画ウィンドウを開く。
・newpen( ) で色を指定する。色番号は 0 – 15 が設定可能で、それぞれの色は
　0:黒 1:白 2:赤 3:緑 4:青 5:シアン 6: マゼンタ 7:黄 8:灰色(暗) 9:灰色
　10:赤(暗) 11:緑(暗) 12:青(暗) 13:シアン(暗) 14:マゼンタ(暗) 15:黄(暗)
・newpen(win, 4) などのグラフィクス関数の先頭にある引数 win はお決まりなので気にしない。
・最後が return 0; で終了している（2 ページ前に説明あり）

・fillrect( ) で長方形を描く。引数は左から描く長方形の位置（左下カドの座標）を x, y の順に、次に幅、高さを示す。（右図）
塗りつぶしの色は fillrect( ) 実行前に newpen( ) で指定した色。



・ggetch( ) でキー入力を待つ。これが無いと「描画ウィンドウを開き、描いて、すぐ消える」ため、結果が見えない。（目に止まらない）
・gclose( ) で描画ウィンドウを閉じる。

□ 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。
変数宣言の float あたりですが、実数の処理については次回以降にやります。
（興味のある受講生は教科書p. 115 以降の 7 章を参照。）

□ 課題 1.

先のサンプルプログラムを修正して、棒の本数を半分、あるいは倍にしてください。

どちらか一方だけ提出すれば良いです。

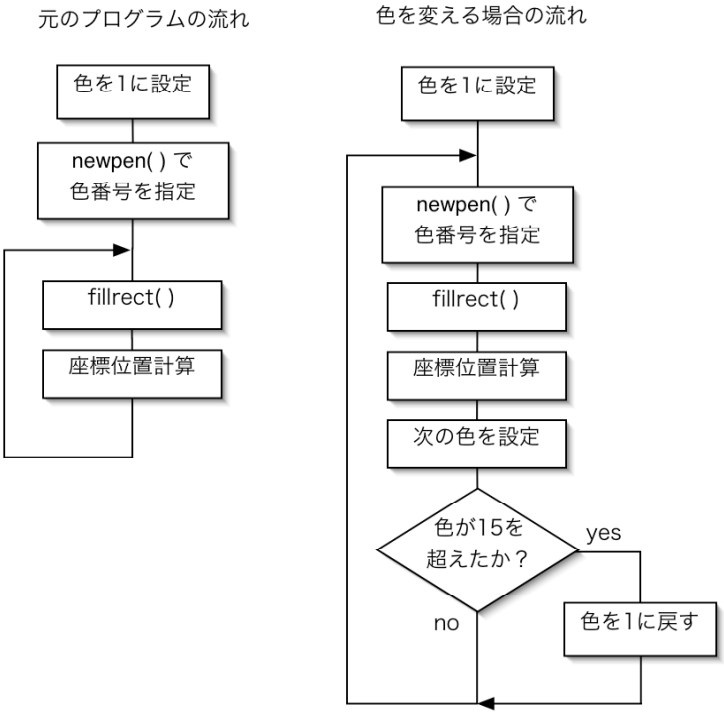
（但し両方試してみること。課題は提出が目的ではないのだから。）

□ 課題 2.

棒の色を一本ずつ変えるようにプログラムを変更してください。

考え方：
・newpen( ) で色を変えるのですが、黒（色番号 0 ）は背景と同じ色で見えないので避ける。
・15 色（ 1 – 15 ）使いきったらまた 1 から始めれば良い。

右図を見て下さい。左側がサンプルプログラムそのまま、右側が棒の色を一本ずつ変える場合の処理の流れです。
（while による繰り返し条件判定の部分は省略しています。）



違いは以下の通りです：
・newpen() 関数の位置が違う
・色番号が固定的でない
・色番号を変化させる処理がある
・ある条件で色番号が戻る