ランダム関数、switch による分岐

■ random 関数

サンプルプログラム random1.c を以下に示す。講師の教材ページなどからとってきて実行せよ。seed （後述）となる数値を入力すれば、ランダムな数値が 10 行表示される。

```
% cp /NF/home/kyoin0/yasuda/kisob/random1.c random1.c
% egg –o random1 random1.c
% ./random1
please set the seed :6723
112993461
705555079
…(中略)…
16087886
2019363506
%
```

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

main()
{
  int seed, i;
  long int r;

  printf("please set the seed :");
  scanf("%d",&seed);
  srand(seed);

  for(i=0;i<10;i++) {
    r=rand();
    printf("%ld¥n", r);
  };

  exit(0);
}
```

ポイント：
- rand() 関数によって乱数が生成されている。rand() とはつまり引数を与えず関数を呼び出すことを意味する。（引数が無い場合でも括弧は省略してはいけない。）
- rand() は呼び出されるたびにランダムな値を返す。値は 2 の 32 乗未満(0から4294967295)の範囲。
- はじめに seed （乱数の種）を入力するよう求められるが、ここに同じ数値を与えると同じ乱数列が表示される。（試しに何度か実行して確認せよ）
- こうした「種」となる数値をもとに生成される乱数を疑似乱数と呼ぶ。

0から4294967295 の範囲で乱数を返されても扱いにくい場合が多い。たとえば 0 から 99 までの乱数が必要になった場合は、rand() 関数が生成する乱数を 100 で割ったあまりを用いると良い。

```
printf(“%d¥n”, r%100 );
```

などと修正して実行すると 0-99 までの範囲でランダムな数が返ってくるのが分かるだろう。

■ 課題 1.

atikoti1.c というプログラムがあるので手元にとってきて実行し、動作を確認せよ。四角をランダムな場所に描くはずである。
このプログラムを修正して、描くたびにランダムな色を使うように変更せよ。終わらないプログラムのままでかまわないが、可能なら 30 個程度描いたら終了するようにしておくと良い。

できあがれば講師に見せて OK が出たものを課題提出システムに登録すること。

■ 課題 2.

今度は四角い枠だけでなく、描くたびに異なる絵柄を描くように変更せよ。
別紙に必要になる（であろう）C 言語での条件分岐についての文法と、画像を描くための関数のサンプルを示す。どのように使うかは各自で工夫して試すこと。

できあがれば講師に見せて OK が出たものを課題提出システムに登録すること。

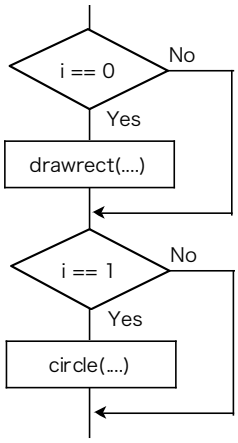
■ if と else 、switch による条件分岐

たとえば i という整数型の変数に 0, 1 のいずれかの数値が入っているとする。0 だったら□を描き、1 だったら○を描くようにする場合、以下のような条件分岐の方法がある。

ポイント：if と else 、その前後の {} によるブロック、行末を示すセミコロン（；）に注意。

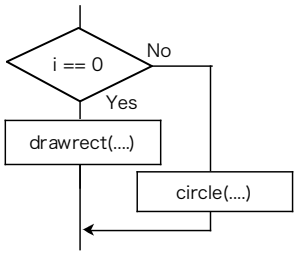
if 文を複数使う。

```
if( i == 0 ) {
  derawrect(....);
};
if( i == 1 ) {
  circle(....);
};
```



else つきの if 文を使う。

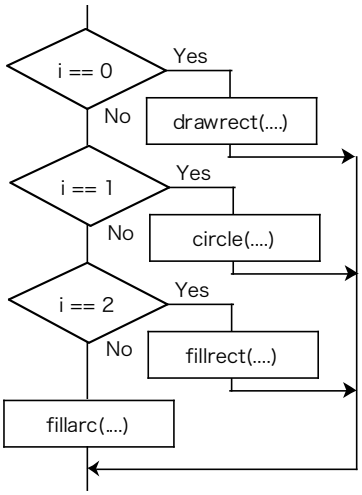
```
if( i == 0 ) {
  drawrect(....);
} else {
  circle(....);
};
```



0 or 1 に限らず、0,1,2,3 のいずれかの数値に応じて、□、○、■ 、●を描くには以下のようにする。

else if を使う。

```
if( i == 0 ) {
  drawrect(....);
} else if( i == 1 ) {
  circle(....);
} else if( i == 2 ) {
  fillrect(....);
} else {
  fillarc(....);
};
```

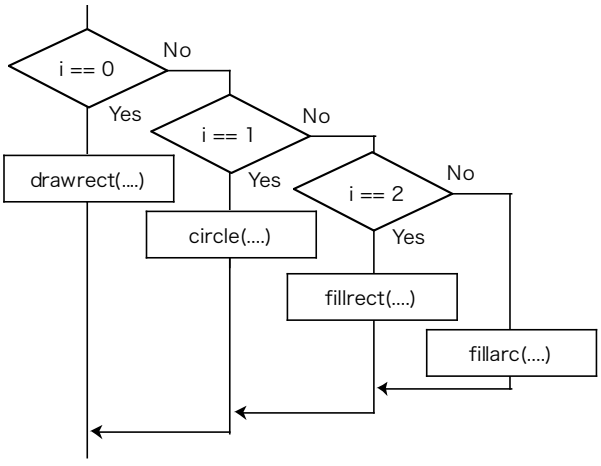


ポイント：
- まず if() によるブロック、次にelse if() によるブロックが複数、最後に else によるブロックという並びに注目。
- else ブロックは省略可。

if 文を以下のように多重に設定することもできる。構造の違いに注目。（各ブロックの範囲を［ で示した。）

if と else を多重に使う。

```
if( i == 0 ) {
  drawrect(....);
} else {
  if( i == 1 ) {
    circle(....);
  } else {
    if( i == 2 ) {
      fillrect(....);
    } else {
      fillarc(....);
    };
  };
};
```

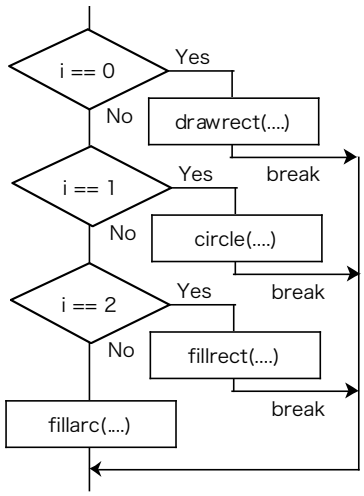


先の例は switch 文を使って以下のようにも書くこともできる。

ポイント：
- switch( ) 文の ( ) 内には i > 2 のような条件式ではなく、評価する変数や計算式を書く。
- case 句にはその変数の値や計算の結果を指定する。
- case 句の終わりには break; を書く。
- どの case にも合致しない場合には default: 句の処理を行う。default: 句は省略可。

```
switch を使う例

switch( i ) {
case 0:
  drawrect(....);
  break;
case 1:
  circle(....);
  break;
case 2:
  fillrect(....);
  break;
default:
  fillarc(....);
}
```



■ EGGX の関数（一部）

□ 四角い枠を描く

```
drawrect(win, x, y, w, h);
```
座標位置 (x, y) から幅 w、高さ h の四角を描く。x, y, w, h いずれも実数型変数。

例：座標位置 (250, 120) を中心とした幅 20, 高さ 10 の長方形の枠を描く。
```
x=250.0; y=120.0;
w=20.0; h=10.0;
drawrect(win, x-(w/2.0), y-(h/2.0), w, h);
```

□ なかを塗りつぶした四角を描く

```
fillrect(win, x, y, w, h);
```
上に示した drawrect() 関数と使い方は同じ。違いは中を塗りつぶすかどうかだけ。

□ 円を描く

```
circle(win, x, y, w, h);
```
座標位置 (x, y) から横幅 w 、高さ h の円を描く。x, y, w, h いずれも実数型変数。

例：座標位置 (250, 120) を中心とした横方向半径 20, 縦方向半径 10 の円を描く。
```
x=250.0; y=120.0;
w=20.0; h=10.0;
circle(win, x, y, w, h);
```

□ 円弧を描く

```
drawarc(win, x, y, w, h, s, e, d);
```
座標位置 (x, y) から横方向半径 20, 縦方向半径 10 の弧を描く。その開始角度は s 度から e 度まで（0-360までの度を与える)。d が 1 なら左回りに描き、-1 なら右回りに描く。x, y, w, h, s, e は実数型変数。ｄのみ整数型。

例：右回りに 30 度から 60 度の位置に（30度ぶんの開き角となる）扇形を描く。
```
drawarc(win, x, y, w, h, 30.0, 60.0, -1);
```

□ なかを塗りつぶした円弧（扇形）を描く

```
fillarc(win, x, y, w, h, s, e, d);
```
上に示した drawarc() 関数と使い方は同じ。違いは中を塗りつぶすかどうかだけ。
s=0.0; e=360.0; としてfillarc()を呼び出すと中が塗りつぶされた円を描くことになる。