基礎プログラミング演習 II 教材 (#11)

■ ループ落ち穂拾い

□ do while 文

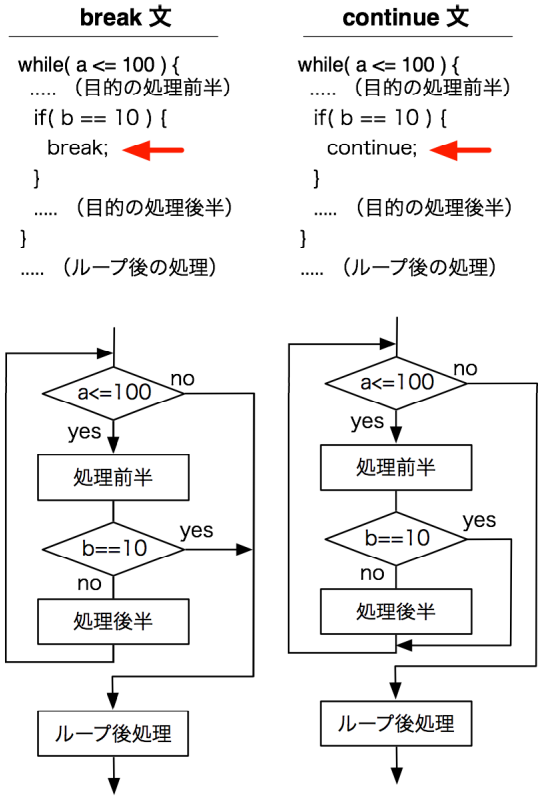
★教科書 p.80 以降参照

ポイント：
・do while の構文と機能
・while との相違（条件判定が後にくること）

□ continue 文：break 以外の実行制御

break 文によって繰り返し制御（while）のブロックから脱出する方法について説明しました。

continue 文によって、繰り返し処理のそれ以降の処理をスキップして、次の繰り返し周期に移ることができます。（興味のある受講生は教科書 p.87 も参照。p.61 には break は if 文にも適用できるとあるがこれは誤り。）



continue を使うケースはそう多くなく、ループの最初に「y が負ならそれ以降の処理は不要」といった除外条件をそれと分かりやすく書く場合などに有効です。

以下に例を示しておきます。左右共に同じ振る舞いをするプログラムですが、continue を使って記述した左の方が、冒頭の if 文のことを考慮しなくてはいけない範囲（矢印でマークした部分）が短くなっています。右側のように記述した場合、if 文の終わりが遠くなり、何十行も間に入ってしまうと分かりにくくなってしまいます。

```
for(i=1; i<=10; i++) {
  if( i%3 == 0 ) {
    continue;
  }

    処理いろいろ

}
```

```
for(i=1; i<=10; i++) {
  if( i%3 != 0 ) {

    処理いろいろ

  }
}
```

■ 多重の if 文

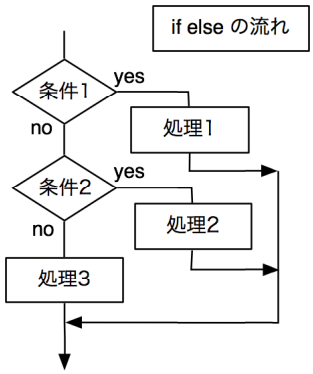
★教科書 p.50 以降を参照して、以下のことを理解する。

・三つ以上の分岐をどのようにして実現するか
・if else 文の構造

□ else if と続けない記述

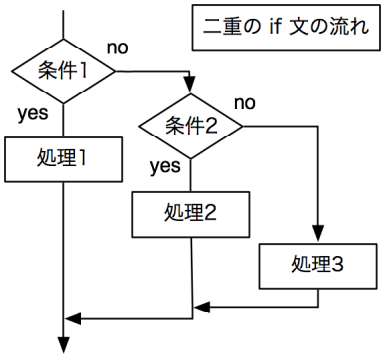
教科書に出てきたのは以下のような記述による多数分岐でした。

```
if(条件 1) {
  処理 1;
} else if(条件 2) {
  処理 2;
} else {
  処理 3;
}
```



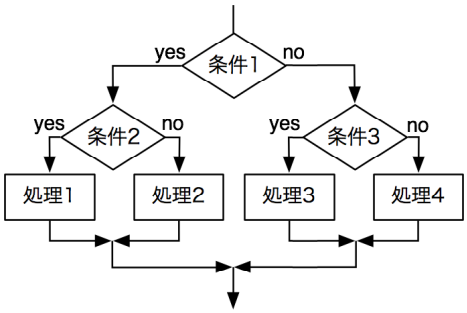
同じ意味の処理を if 文を入れ子にして表現することもできます。
入れ子も何重にでもできます。

```
if(条件 1) {
  処理 1;
} else {
  if(条件 2) {
    処理 2;
  } else {
    処理 3;
  }
}
```



このように else でない側を入れ子にすることも可能です。

```
if(条件 1) {
  if(条件 2) {
    処理 1;
  } else {
    処理 2;
  }
} else {
  if(条件 3) {
    処理 3;
  } else {
    処理 4;
  }
}
```



□ 課題 1.

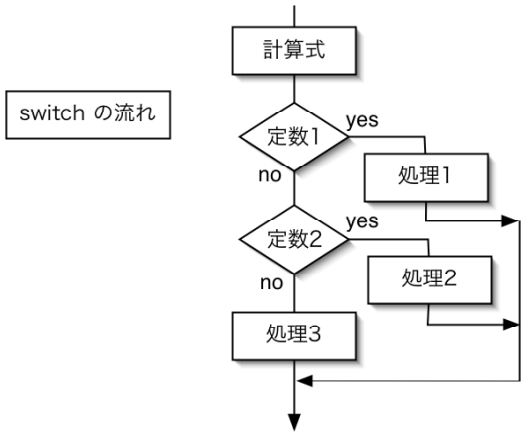
前回の課題 3.のプログラムを加工し、描く図形を 4 種類指定できるようにしてください。あと二つの図形には（中を塗りつぶさない）四角を描く `drawrect( )` 、（中を塗りつぶさない）円を描く `circle( )` を利用すると良いでしょう。各関数の仕様については<<プログラミングガイド 9.3>> を参照。

■ switch 文による場合分け

★教科書 p.57 を参照して、switch 文の機能を理解する

書式：

```
switch( 式 ) {
case 定数 1:
  処理 1;
  break;
case 定数 2:
  処理 2;
  break;
..（略）..
default:
  処理 n;
}
```



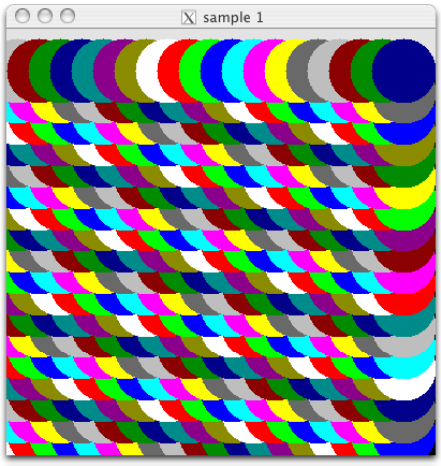
注意点：
・case に続く条件には整数の定数しか書けません。（x < 10 等条件式は不可。1+2 等定数式は可。）
・case による分岐は幾つでも書けます。
・どのケースにも該当しなかった場合は default: に続く処理を実行します。
・default は省略できます。（その場合該当しないケースでは何もせずに swtich を通過します。）
・各処理の最後には break; を付けてください(*1)。break はループを脱出するものとして登場していましたが、switch からの脱出にも使います。
・switch の直後の { と、対応する } を忘れないように。

*1 興味のある受講生は教科書 p.60 以降を参照

□ 課題 2.

課題 1. を、switch を用いて書き直してください。

□ 課題 3.



・type だけでなく、長さも入力し、
・四角形の辺の長さ、円の径をその長さで描画する。

つまり以下のような実行時の入力によって右図のような結果となるようにしてください。
（プロンプトに続いて 2 30.0 と入力した。）

```
$ ./switch2
drawing type (1-4) and size = 2 30.0
$
```

但し double 型変数への scanf () によるデータ入力は次に述べる点についての注意が必要です。

□ double 型変数への scanf

EGGX の関数などで説明したように、関数の引数が実数であった場合に、float と double の型合わせを厳密に守る必要はありません。しかし scanf( ) では事情が異なります。

scanf( ) 関数で実数型に値を入力する場合、float 型変数であれば %f、double 型であれば %lf を変換文字として指定します。（つまり明確に型を合わせなければなりません）

```
float f; double d;
scanf(“%f %lf”, &f, &d);
```

教科書 p.122 の表 7.3 及び脚註には printf と scanf で変換文字は同様に機能するように読めますが(p.121 最終パラグラフも)、そうではありません。
scanf では float は %f、double は %lf でなければなりません。
printf では float, double 共に %f, %lf が同様に機能します。（どちらでも構いません）

scanf でだけ厳密に合わせる必要があるのは、まだ学んでいない & つまり「ポインタ」の構造・機能に起因するものです。このクラスでは細かく説明しませんので、ポインタについて学ぶまでは「double の scanf には %lf」と丸覚えしてください。