switch 文制御

多分岐

C/C++、Java プログラム経験者であれば、これまでの内容は退屈なものだったでしょう
switch 文もまた、C/C++、Java など多くの言語に共通する使用です
しかし、C# の switch は C/C++、Java とは異なる使用を含むので
この章は C/C++、Java プログラム経験者にもよく読んでほしい

if - else 文は2者択一の2分岐フロー制御が基本でした
ある特定の式の結果を複数に分岐させたい場合は、これよりも **switch** 文が有効です

```
switch (expression)
{
case constant-expression:
 statement
 jump-statement
[default:
 statement
 jump-statement]
}
```

expression には、評価する式を指定します
ここは if 文と違い、**整数型、char、string、enum**型のいずれかでなければいけません
（enum は列挙型と呼ばれるもので、これについては後記します）

constant-expression には、expression と比較する定数式を指定します
この定数式と expression が等しいときにその case の statement が実行されます
statement には、1つ以上のステートメントを指定することができます
default は、全ての case に条件が合わなかった時に実行するブロックです
default は省略可能で、その場合で条件が合わなければ switch ブロックを抜けます

jump-statement には、switch ステートメントから抜け出るためのステートメントを指定します
基本的には switch ステートメントは **break** ステートメントで終了します
break は現在実行中の switch 文やループ制御などを抜け出すステートメントです
jump-statement にはこのような switch を抜け出せるステートメントを指定します

```
class Test {
 static void Main() {
  int msg = 1;
  switch(msg) {
  case 0:
   System.Console.WriteLine("Di Gi Charat");
   break;
  case 1:
   System.Console.WriteLine("Kitty on your lap");
   break;
  case 2:
   System.Console.WriteLine("Tokyo mew mew");
   break;
  default:
   System.Console.WriteLine("Nekoneko Zoo");
   break;
  }
 }
}
```

変数 msg は何らかの意味のある定数を受ける変数だと想定します
その意味のある値（メッセージ）は多種多様に用意されているならば
if で制御するよりもこのように switch 文で分岐させたほうがはるかに効率がよいのです
msg 変数の値を変更してコンパイルし動作を確認して下さい

ところで、C/C++ 経験者ならば default 区の最後に break を指定することは奇妙に思うかもしれない
C/C++ 言語では、case が実行されると break が発見されるまで実行するのですが
このとき break がなければ次の case までが継続して実行されるという仕様でした
これは**フォールスルー**と呼びますが、C#ではフォールスルーは発生しません

一般にフォールスルーが使われることはほとんどなく
何十万行に及ぶソースを書いても、フォールスルーが見られるのは数える程度です
そこで、C# はメリットよりもフォールスルーによるバグのリスクの方が高いとして
各 case 及び default 区の末尾に何らかのジャンプステートメントがなければ
フォールスルーではなく**コンパイルエラー**を発生される仕様を選んだのです

```
class Test {
 static void Main() {
  int msg = 10;
  switch(msg) {
  case 0:
   System.Console.WriteLine("Di Gi Charat");
  case 1:
   System.Console.WriteLine("Kitty on your lap");
  case 2:
   System.Console.WriteLine("Tokyo mew mew");
  default:
   System.Console.WriteLine("Nekoneko Zoo");
  }
 }
}
```

このプログラムは case や default の末尾にジャンプステートメントがありません
そのため、コンパイル時にエラーが発生してしまいます
例えばこれを Java と見たてれば、case 0 のステートメントリストが実行されれば
以下の全ての case や default も実行されるということになります

それと、C# の switch 文は文字列型を使用することもできます
関係演算同様に長さと各位置の文字が全て等しければ実行されます

```
class Test {
 static void Main() {
  string str = "Di Gi Charat";
  switch(str) {
  case "Tokyo mew mew":
   System.Console.WriteLine("ご奉仕するニャン");
   break;
  case "Di Gi Charat":
   System.Console.WriteLine("目からビーーーム！");
   break;
  default:
   System.Console.WriteLine("Kitty on your lap");
   break;
  }
 }
}
```

実行すると case "Di Gi Charat" が実行されます
大文字と小文字も区別するという点で注意してください

---

switch

```
switch (expression)
{
case constant-expression:
 statement
 jump-statement
[default:
 statement
 jump-statement]
}
```

式を評価して一致する定数式のブロックを実行します

expression - 評価する式を指定します
constant-expression - expression と比較する定数式を指定します
statement - 1つ以上のステートメントを指定します
jump-statement - ジャンプステートメントを指定します

前のページへ　戻る　次のページへ