if 文制御

条件分岐

プログラムは、常に逐次実行するだけではありません
状況に応じて様々な位置に分岐して、バラエティのあるプログラムを作ってみたいものです

プログラムの分岐の基本は **if** ステートメントで行います
以前「ブーリアン」で説明した true と false という論理値がここで役に立ちます

if (expression) statement

expression には bool 型の式を指定します
この式が **true の時は statement を実行** します
false であれば、statement を実行せずに次のステップに移行します

このようなことから、一般に expression は関係演算が中心になります
C/C++ はブーリアンを 0 ならば偽、それ以外ならば真と判断していましたが
C# は型として独立しているため、expression に数値型を指定できないことに注意してください
true や false を直接指定することもできますが、リテラルを指定することに意義はありません

```
class Test {
 static void Main() {
  int x = 10;
  if (x <= 100) System.Console.WriteLine("Kitty on your lap");
  if (x > 100) System.Console.WriteLine("Tokyo mew mew");
 }
}
```

このプログラムは、変数 x が 100 と同じかそれ以下の場合と
100 より大きい場合で表示する文字列が変わるというプログラムです
この状態では Kitty on your lap が表示されますが
変数 x の初期値を 101 以上にすると Tokyo mew mew が表示されるようになります

if は自然言語に翻訳するならば「もし…ならば…する」という動作のステートメントです
これに「そうでなければ…する」という動作を追加させたい場合
前方の if の条件を否定するという方法もありますが、確実な方法ではなくリスクを含みます

確実に if の条件が false だった時に特定のステートメントを実行したい場合
if に関連付けた **else** ステートメントを使用します

if (expression) statement1
[else statement2]

これが正式な if 文の全容です
expression にはブール式を指定し、statement1 は
expression が true の時に実行するステートメントを
statement2 には、expression が　false だった時に実行するステートメントを指定します
else 文は上記のプログラムのように省略することが可能です

```
class Test {
 static void Main() {
  bool bl = false;
  if (bl) System.Console.WriteLine("Kitty on your lap");
  else System.Console.WriteLine("Tokyo mew mew");
 }
}
```

変数 bl が false のため if ステートメントでは偽と評価されます
そのため、if 文に関連付けられた else 文のステートメントが実行されます
このように else 文は **if に関連付けられていなければなりません**
else の前に if 文が見つからない場合はコンパイルエラーとなります

複数の else を重ねて if ステートメントによる多分岐を実現することもできます
実際に複雑な処理をするプログラムの開発にはよく用いられる手法です

```
class Test {
 static void Main() {
  byte x = 2;

  if (x == 0) System.Console.WriteLine("Kitty on your lap");
  else if (x == 1) System.Console.WriteLine("Di Gi Charat");
  else if (x == 2) System.Console.WriteLine("Tokyo mew mew");
  else System.Console.WriteLine("Magical nyan nyan TARUTO");
 }
}
```

このプログラムは x の値が 0、1、2、それ以外の 4 パターンに分岐します
x の値をそれぞれの値に変更してコンパイルすると確認できるでしょう

プログラムを見ると各 else は前方の if にそれぞれ対応しています
最後の else 文は全ての条件が偽だった時に実行することになります

なお、if のような他のステートメントを持つステートメントを**複合ステートメント**と呼び

ステートメントに包含されるステートメントを**埋め込みステートメント**と呼ばれます
if や else で指定するステートメントは埋め込みステートメントになります
この埋め込みステートメントは通常のステートメントと異なり**宣言はできません**

```
class Test {
 static void Main() {
  bool bl = true;
  if (bl) int x = 10; //エラー
 }
}
```

このプログラムは if の埋め込みステートメントで変数を宣言しているためコンパイルできません
特定の分岐で変数を宣言したり、後に説明する「ラベル」を指定するには
後記するブロックのステートメントとして行う必要があります

ブロック

if - else ステートメントは条件があう時に単一のステートメントを実行するものです
しかし、複雑な処理の場合は複数行の処理にまたがることがあります

if - else は、複数行のステートメントを実行する能力はありません
そこで**ブロック**という方法で複数のステートメントを1つにまとめます
ブロックはこのような、1つのステートメントが許可される文脈で
複数のステートメントを記述できる非常に便利な機能です

{ statement-list }

statement-list には、1つ以上のステートメントを記述します
ブロックが実行されると制御は statement-list に移行し
statement-list の実行が終了すると、制御を戻します

```
class Test {
 static void Main() {
  int x = 0 , y = 100;

  if (x < y) {
   x++;
   y--;
   System.Console.WriteLine("x = " + x + " : y = " + y);
  }
 }
}
```

このプログラムの if 文は真と評価された場合
ブロックに制御を移して、ブロック内のステートメントリストが実行されます
このように、制御文など単一のステートメントしか扱えない文脈の中には
ブロックを指定することによって複数のステートメントを実行できるようになります

条件演算子

if は一つのステートメントとして分岐処理をサポートしていますが
1行単位のごく単純な分岐処理の場合、条件演算子を使うことがあります

条件演算子はステートメントではなく演算子としてブーリアンを評価し適切な値を返します
if ステートメントと異なり、一つの式の中に指定することができます
条件演算子は **?:** をつかって表現します

cond-expr ? expr1 : expr2

cond-expr には、ブーリアン型の条件式を指定します
expr1 は、cond-expr が true だった時に評価する式を
expr2 は、cond-expr が false だった時に評価する式を指定します

演算子は型を明示しなければいけないため、expr1 と expr2 の型は互換性が必要です
双方が同じ型であれば、その型がこの式の型となりますが
expr1 から expr2 への暗黙の変換が可能で、expr2 から expr1 への変換が存在しない場合 expr2 が
その逆ならば expr1 がこの条件式の型と判断されます
どちらからも暗黙の変換ができない場合はコンパイルエラーが発生します

```
class Test {
 static void Main() {
  bool bl = true;
  System.Console.WriteLine(bl ? "Kitty on your lap" : "Tokyo mew mew");

  bl = false;
  System.Console.WriteLine(bl ? "Kitty on your lap" : "Tokyo mew mew");
 }
}
```

評価する条件式によって、条件演算子が返す値が変化することを確認できます
expr1 と expr2 の型が同じなので、この条件式は「文字列型」とコンパイラは判断します

---

if - else

if (expression) statement1

[else statement2]

ブール式を判定し、評価にしたがって指定されたステートメントを実行します

expression - ブール式を指定します
statement1 - expression が true のときに実行するステートメントを指定します
statement2 - expression が false のときに実行する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