

Selection and repetition statements in C#

Selection - constructions

- If
- switch

Repetition / iteration / loops - constructions

- while
- do while
- for
- foreach (for iteration over collection and arrays)

Selection statements in C#

If statement

Basic syntax:

```
if ( boolean-expression )  
    if-statement ;  
  
else  
    else-statement;
```

The else section is optional.

The if-statement and else-statement may be a compound statement

Compound statement is

```
{  
    statement-1;  
    statement-2;  
    ....  
    statement-n;  
}
```

Selection statements in C# (continued)

If and else statements may be nested.

For multiselection you can use the short nested form.

```
if ( boolean-expression-1 )
    if-1-statement;
else if ( boolean-expression-2 )
    if-2-statement;
.....
else if ( boolean-expression-n )
    if-n-statement;
else
    else-n-statement;
```

For the nested if and else the statement could also be a compound statement {.....}

Selection statements in C#

Switch statements

```
switch ( switch-expression )  
{  
    case case-1-value : case-1-statement-1;  
                       case-1-statement-n1;  
                       break ;  
    case case-2-value : case-2-statement-1;  
                       case-2-statement-n;  
                       break ;  
    default :          default-statement-1;  
                       default-statement-n;  
                       break ;  
}
```

Switch-expression can be int, boolean, char or string (so come on java and C++)

Case-value must be of the same type as the switch-expression.

You may exclude all the statements for the case, if you want to fall through to the next case or for the last case to the default.

Repetition statements in C#

while-loops

```
while ( boolean-expression )  
    while-statement ;
```

Loops while the boolean-expression is true.

The while-statement may be a compound or an empty statement.

You can break out of a while-loop by the "break;" statement.

do-while-loops

```
do  
    do-while-statement ;  
while ( boolean-expression );
```

Loops while the boolean-expression is true.

The do-while-statement may be a compound or an empty statement.

You can break out of a do-while-loop by the "break;" statement.

Repetition statements in C#

for-loops

```
for ( init-statement ; boolean-expression ; in-/decremental-statement )  
    for-statement ;
```

Loops while the boolean-expression is true.

The init-statement is done once before loops.

The in-/decremental-statement is done at end of each loop.

The three statements may be a compound or an empty statement.

Normally only the for-statement are a compound statement.

You can break out of a for-loop by the "break;" statement.

The for-statement can be seen as a compact while construction. The only difference is that variables declared in the init-statement, don't live after the loop has ended.

Repetition statements in C#

foreach-loops

```
foreach ( foreach-class foreach-object in foreach-collection )  
    foreach-statement ;
```

Loops while there are an foreach-object in the foreach-collection.
Each object in the collection must be of foreach-class, as they are typecasted to the foreach-class.

The foreach-statement may be a compound.

You can break out of a foreach-loop by the "break;" statement.

The foreach-collection must implement the IEnumerable interface.

In the System.Collections namespace library you find a lot of collections that can be uses.

You can use C# array's of one or more dimension.