FUNDAMENTALS OF PROGRAMMING WITH C#

CONDITIONALS

Liu Fan

isslf@nus.edu.sg

Total:22

Objectives

- Write a program that with alternative execution path in C#

Agenda

- Conditional Statements
  - *if …*
  - *if … else …*
- Single Line vs Multi Line statements
  - statement blocks
- Nested *if* s
- switch - case Statement

Definition of Conditional Statements

- Conditional Statements allow us to execute code selectively. So during a specific situation we can choose to execute some code and not others.

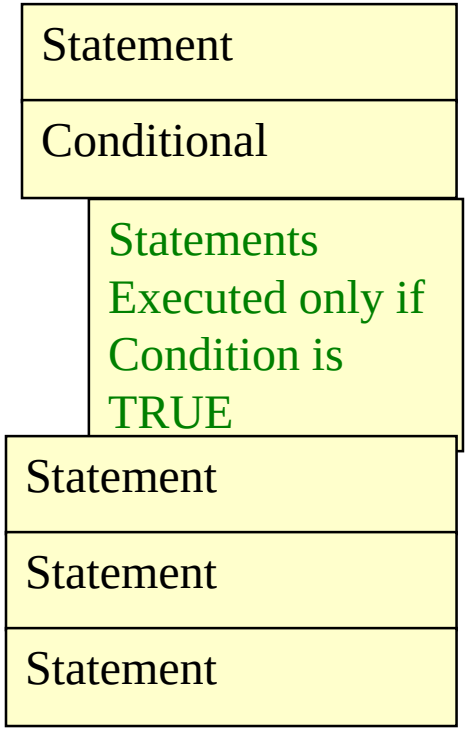
- The situation is identified by a comparison expression (or a more complicated logical expression).

- When the comparison returns a TRUE, a selected portion of code is executed.

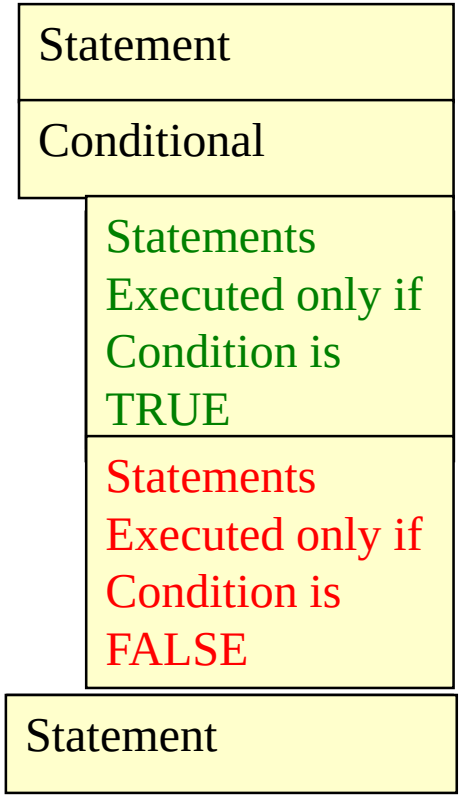
- There are also variations that allow us to execute some code if the comparison is TRUE, and other code if the comparison is FALSE.

Execution Based on Condition

if Statement



if … *else* Statement



IF statement - The Syntax

- The *if* statement in C# has a specific form.
  - The following is an example of a Single *if* statement:

This statement executes irrespective of whether the "if" condition is true or false.

if (condition **)**

This Statement would Execute ONLY when the "if " condition is true **;**

This Statement Executes irrespective of whether the "if" condition is true or false.

Note:

Statements in **block (& this colour)** are required by the language.

You can write statements in place of lines indicated as *italics (& this colour)* for actions to be performed when the condition is true. The statements indicated as *italics (& this colour)* would be executed irrespective of whether the condition is true or not.

IF statement - The Syntax

- The *if* statement in C# can have several statements executed in sequence when the condition is true.
  - This is called multi line *if* statement and has to be a "statement block":

```
This statement executes irrespective of whether the "if" condition is true or false.
if ( condition )
{
   This Statements to Execute ONLY when the "if " condition is true;
   This Statements to Execute ONLY when the "if " condition is true;
   ... ...
  This Statements to Execute ONLY when the "if " condition is true;
}
This Statement Executes irrespective of whether the "if" condition is true or false.
```

Note:

Statements in **block (& this colour)** are required by the language.

You can write statements in place of lines indicated as *italics (& this colour)* for actions to be performed when the condition is true. The statements indicated as *italics (& this colour)* would be executed irrespective of whether the condition is true or not.

Examples of the *if* Statement

```
string letter_grade;
int grade;
…
if (grade >= 80)            // if the grade is greater than or equal to 80 then
   letter_grade = “A”;      // the letter grade is an “A”
```

```
int hours, days;
…
…
if (hours >= 24)            // if the hours are 24 or more
{                           // opening braces indicate beginning of if block
   days = days + 1 // increase the number of days by one
   hours = hours - 24       // set the hours in the 0 to 24 range
}                           // closing braces indicate end of if block
Console.WriteLine("Days: {0}   Hours: {1}", days,hours);
```

Quiz

- Given the following program, what is the output?

```
double A;
A = 5;
if (A > 5) {
  System.Console.WriteLine(A);
}
```

a) The number 5

b) There is no output

IF … ELSE statement The Syntax

- The ***if … else*** statement in C# has the form:

```
This statement executes irrespective of  whether the "if" condition is true or false.
if (  condition  )
{
   This Statements Executes ONLY when the "if " condition is true;
   … …
  This Statement Executes ONLY when the "if " condition is true;
}
else
{
 This Statement Executes ONLY when the "if " condition is false;
   … …
  This Statement Executes ONLY when the "if " condition is false;
}
This Statement Executes irrespective of  whether the "if" condition is true or false.
```

Example of *if … else* Statement

```
using System;

class Example_If_Else
{
  static void Main()
  {
     double PI;
     Console.WriteLine("What is the value of PI nearest to four decimal points?");
     PI = Convert.ToDouble(Console.Readline());
     if ((PI > 3.1414) && (PI < 3.1416))
          Console.WriteLine("Very good!");
     else
          Console.WriteLine("Sorry, the answer is approximately 3.1415");
   }
}
```

Blue is executed always

Green is executed when "if condition is true"

Orange when the "if-condition" is false

Examples of the *if … else* Statement

- Consider the Main program to compute roots of the quadratic equation using the standard algebraic formula

```
static void Main()
{  double r1, r2, a, b, c, D;
   a=Convert.ToDouble(Console.ReadLine());
   b=Convert.ToDouble(Console.ReadLine());;
   c=Convert.ToDouble(Console.ReadLine());;
   D = (Math.Pow(b,2)-4*a*c);
   Console.WriteLine(D);
   if (D< 0)
      Console.WriteLine( "The roots are imaginary");
   else
   {
       r1 = (-b + Math.Sqrt(D)) / (2 * a);
       r2 = (-b - Math.Sqrt(D)) / (2 * a);
       Console.WriteLine("The roots are {0} and {1}",r1, r2);
   }
  Console.WriteLine("Thanks for using this program");
}
```

Blue is executed always

Green is executed when "if condition is true"

Orange when the "if-condition" is false

Blue is executed always

Nesting If Statements

- *"if"* statements can be placed inside one another.
- Recognise that doing this builds a complex logical expression....
- To get to the innermost execution area, all the conditions must be true... Which is precisely the same as *And*'ing them all together!
- Example

```
if (X < Y)
    if (X < Z)
        if (X < A)
            X = 0;      /* This statement will only be executed if X
                           is less than Y,Z, and A, i.e.
                           ( (X < Y) && (X < Z) && (X < A) )   */
```

Nested if with else

- Placing an *else*:
  - If an *else* clause is placed in a nested *if* statement, the *else* operates on (or becomes part of) the innermost *if* statement.
  - Example:

```
if (X < Y)
   if (X < Z)
      if (X < A)
            X = 0;      /* This statement will only be executed if X
                           is less than Y,Z, and A, i.e.
                           ( (X < Y) && (X < Z) && (X < A) )    */
      else
            X = 1;      /* This statement will only be executed if X
                           is less than Y and Z and X is greater than or
                           equal to A, i.e.  (  (X < Y) && (X < Z) && ! (X < A) )  */
```

  - How do we make *else* part of an earlier (outer) *if* statement?

Nested if with else

- Making else to be a part of an outer if:
  - This could be done by forcing the inner if statement inside a block
  - Example:

```
if (X < Y)
    if (X < Z)
    {
        if (X < A)
                X = 0;      /* This statement will only be executed if X
                               is less than Y,Z, and A, i.e. ( (X < Y) && (X < Z) && (X < A) )  */
    }
    else
        X = 1;              /* This statement will only be executed if X
                               is less than Y and not less than Z ; i.e.  (  (X < Y) && ! (X < Z))  */
```

  - The braces mark the beginning and end of an *if* block. The *else* is therefore forced to act on the outer *if*.

Quiz

- When will X be set to 1?

```
{
  if (A)
  {
    if (B)
    {
      if (C)
        X = 0;
    }
  }
  else
    X = 1;
}
```

A: When A, B, and C are true
B: When A and C are true and B is false
C: When A and B are true and C is false
D: Answer B and C
E: When A is True
F: When A is False and B & C are False
G: When A is False
H: Answer F and G
I: None of the above

This one is a pain

- Write a program that would print “Sun” if Day=1, “Mon” if Day=2 …… “Sat” if Day=7.
- One possible way is:

```
int Day;
Day = Convert.ToInt32(Console.ReadLine());
if (Day == 1) Console.WriteLine( “Sun”);
if (Day == 2) Console.WriteLine( “Mon”);
if (Day == 3) Console.WriteLine( “Tue”);
if (Day == 4) Console.WriteLine( “Wed”);
if (Day == 5) Console.WriteLine( “Thu”);
if (Day == 6) Console.WriteLine( “Fri”);
if (Day == 7) Console.WriteLine( “Sat”);
```

- This is inefficient. Why ?

Many if example continued

- I want to improve efficiency. But look, what a mess I've created =>

```
DAY = Convert.ToInt32(Console.ReadLine());
if (DAY == 1)
     Console.WriteLine( "Sun");
else if (DAY == 2)
     Console.WriteLine( "Mon");
else if (DAY == 3)
     Console.WriteLine( "Tue");
else if (DAY == 4)
     Console.WriteLine( "Wed");
else if (DAY == 5)
     Console.WriteLine( "Thu");
else if (DAY == 6)
     Console.WriteLine( "Fri");
else if (DAY == 7)
     Console.WriteLine( "Sat" );
else
    Console.WriteLine("Out of Range");
```

The Switch Case Statement

- The switch…case Statement is used instead of the If Statement when the decision is between many alternatives instead of two.
- The switch…case decides which block of code to execute based on an expression.
- The expression might be, for instance, a number. Each case might then be a potential value for that number. When the number of the expression matches the case, that case is selected.
- Normally in *C or C++ or Java or other languages* all the cases are tried (even if an earlier match is found). THIS IS NOT TRUE WITH C#.
  - You have to use a "break" statement to terminate the evaluation of other cases at the end of each case block. You may use a goto statement instead. Else a compilation error occurs.
  - By forcing a break at each case, the switch…case construct strictly acts like the else…if construct.
- Sometimes its difficult to know all the cases that can come up.
  - If we had used the if… else if… construct we could have used an else at the end to cover other possibilities.
  - The switch… case gives us this facility by the use of "default:" keyword. This essential says, “Hey, if nothing else matches, do this!”

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/switch

SWITCH - CASE
The Syntax

```
switch (Test Expression )
{
  case Value_1 :
    First Statement Block
        break;
  case Value_2 :
    Second Statement Block
        break;
          -
          -
          -
  default:
    Default Statement Block
     break;
}
```

The Test Expression can be an integer or string expression.

Note that it is not a condition but a value (or expression).

The Result Value List is either a string or integer corresponding to the test expression.

The use of default block is advisable but *not mandatory!*

In this structure, all case blocks end with a break (including the default block).

Examples of the Switch Case Statement

```
int Day = Convert.ToInt32(Console.ReadLine());
switch (Day)
{
    case 1:
          Console.WriteLine("Sun");
          break;
    case 2:
          Console.WriteLine("Mon");  break;
    case 3:
         Console.WriteLine("Tue");  break;
    case 4:
         Console.WriteLine("Wed"); break;
    case 5:
         Console.WriteLine("Thu"); break;
    case 6:
         Console.WriteLine("Fri"); break;
    case 7:
         Console.WriteLine("Sat"); break;
    default:
         Console.WriteLine("Out of Range"); break;
 }
```

Summary

- `if` statement can be used for a conditional execution of code
- `if … else …` can be used when there are two mutually exclusive conditions
- Nested `if` is useful when we are checking for multiple conditions
  - This is quite general
- `switch – case` is used when we have multiple equality conditions