C# Conditions & Loops

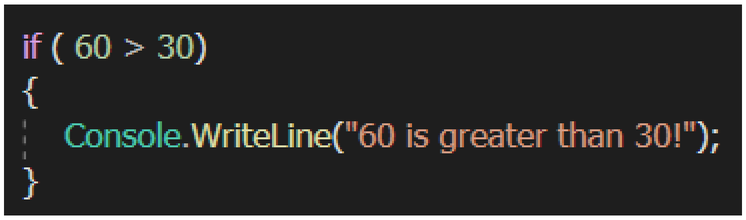
if Statements

- ➔ Allows you to specify a block of code to run if the condition is **True**
- ➔ "if" must always be lowercase

Syntax

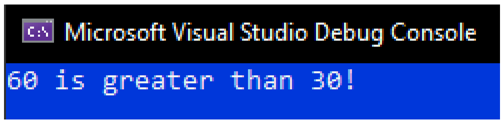
```
if (condition)
{
        // the code in here will be executed if the condition is True
}
```

Syntax Example



```
if ( 60 > 30)
{
    Console.WriteLine("60 is greater than 30!");
}
```

Output:



```
60 is greater than 30!
```

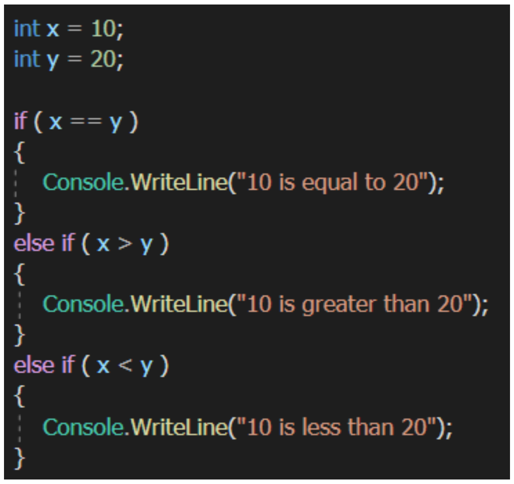
else if Statement

- ➔ Multiple else if statements can be used after the if statement to **add additional condition statements**
- ➔ Will only run when the if statement evaluates to **False**

Syntax

```
if (condition1)
{
        // the code in here will be executed if the condition is True
}
else if (condition2)
{
        // the code in here will execute when:
        //        condition1 evaluates to False
        //        condition2 evaluates to True
}
else if (condition3)
{
        // the code in here will execute when:
        //        condition1 evaluates to False
        //        condition2 evaluates to False
        //        condition3 evaluates to True
}
```

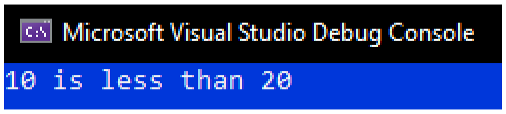
Syntax Example



```
int x = 10;
int y = 20;

if ( x == y )
{
   Console.WriteLine("10 is equal to 20");
}
else if ( x > y )
{
   Console.WriteLine("10 is greater than 20");
}
else if ( x < y )
{
   Console.WriteLine("10 is less than 20");
}
```

Output



```
10 is less than 20
```

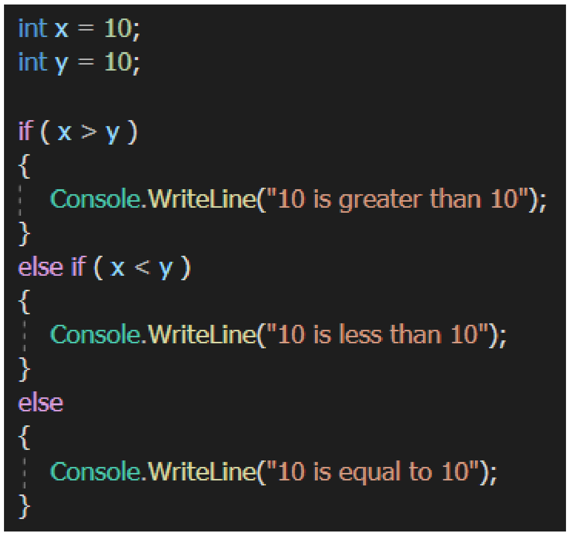
else Statement

- ➔ Use the else statement after the if statement to specify a block of code to run if the condition is **False**
- ➔ Can only come after if or else if statements
- ➔ Will only be used **once** in the if-else statements
- ➔ Cannot contain any conditional statements

Syntax

```
if (condition1)
{
        // the code in here will be executed if the condition is True
}
else if (condition2)
{
        // the code in here will execute when:
        //        condition1 evaluates to False
        //        condition2 evaluates to True
}
else
{
        // the code in here will execute when both condition1 and condition2 evaluate to False
}
```

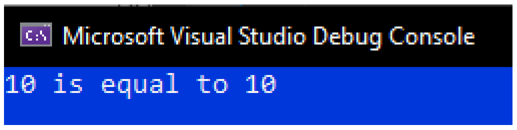
Syntax Example



```
int x = 10;
int y = 10;

if ( x > y )
{
    Console.WriteLine("10 is greater than 10");
}
else if ( x < y )
{
    Console.WriteLine("10 is less than 10");
}
else
{
    Console.WriteLine("10 is equal to 10");
}
```

Output



```
10 is equal to 10
```

switch Statement

- ➔ An alternative to if else statement
- ➔ Like a list of possibilities that have actions for each possibility
- ➔ switch expression is only evaluated **once**
- ➔ default keyword specifies block of code to run if there is no case match
- ➔ break keyword breaks out of the switch block

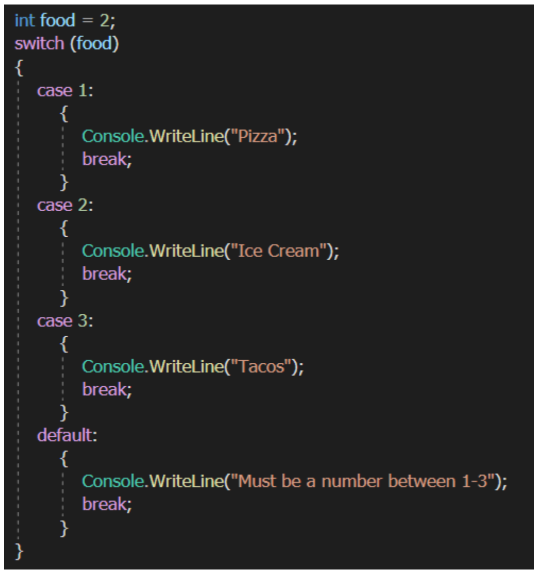
How it works:

1) Value of the expression is compared with the values of each case
2) If there is a match, the block of code inside the case will execute

Syntax

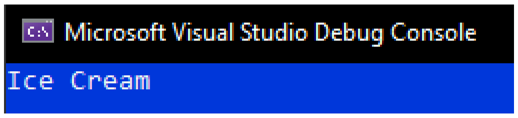
```
switch (expression)
{
    case a:
            // code block here
            break;
    case b:
            // code block here
            break;
    case c:
            // code block here
            break;
    default:
            // code block here
            break;
}
```

Syntax Example



```
int food = 2;
switch (food)
{
    case 1:
        {
            Console.WriteLine("Pizza");
            break;
        }
    case 2:
        {
            Console.WriteLine("Ice Cream");
            break;
        }
    case 3:
        {
            Console.WriteLine("Tacos");
            break;
        }
    default:
        {
            Console.WriteLine("Must be a number between 1-3");
            break;
        }
}
```

Output



```
Ice Cream
```

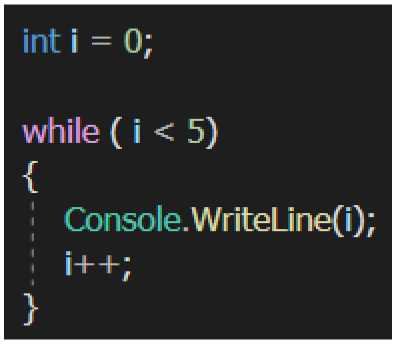
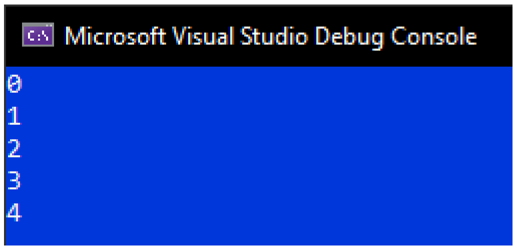
while Loop

- ➔ while loop will loop through a block of code as long as the specified condition is **True**
- ➔ Good loop to choose if you're **unsure** how many times you'll have to loop through

Syntax

```
while (condition)
{
        // code block here
}
```

Syntax Example

```
int i = 0;

while ( i < 5)
{
    Console.WriteLine(i);
    i++;
}
```

Microsoft Visual Studio Debug Console

```
0
1
2
3
4
```

do/while Loop

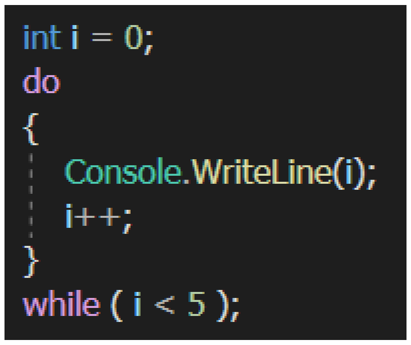
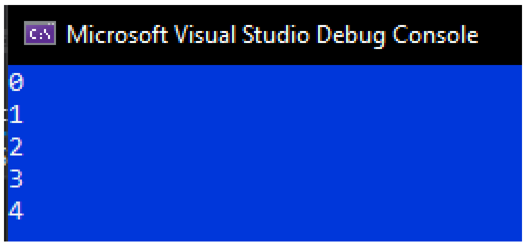
➔ do/while is a variant of the while loop

➔ The code block will execute **once** **<u>before</u>** checking if the condition is **True.** The loop will repeat as long as the condition remains **True.**

➔ The do/while loop will always be executed **at least once** because the code block is executed **<u>before</u>** the condition is tested

Syntax

```
do
{
    // code block here
}
while {condition}
```

Syntax Example	*Output*
	

```
int i = 0;
do
{
    Console.WriteLine(i);
    i++;
}
while ( i < 5 );
```

Microsoft Visual Studio Debug Console

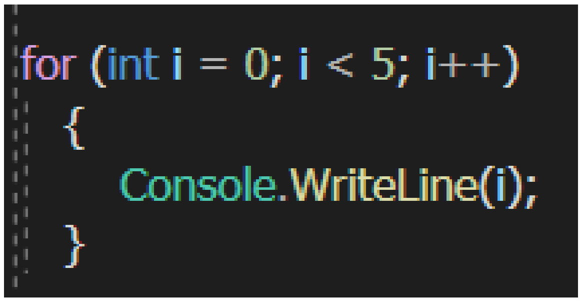
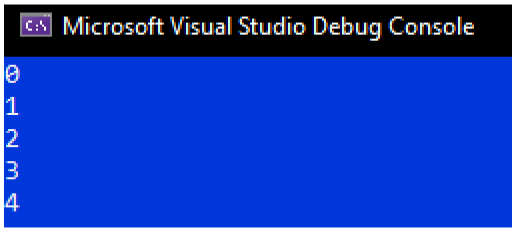
```
0
1
2
3
4
```

for Loop

- ➔ For loops are used when you know **exactly how many times** you want to loop through a block of code

Syntax	*Syntax details*

for (statement 1; statement 2; statement 3)
{
    // code block here
}
	Statement 1 -> executed once before the execution of the code block
	Statement 2 -> defines the condition
	Statement 3 -> executes every time after the code block has been executed

Syntax Example	*Output*
 `for (int i = 0; i < 5; i++) { Console.WriteLine(i); }`	 Microsoft Visual Studio Debug Console 0 1 2 3 4