Selection, Iteration, Sequence and Function

Selection –

Selection allows the developers to create a conditional statement which will require that condition to be met before the execution. For example if the kettle is boiling then we can turn it off not before or event when it's over boiling. We can use the similar theory for other situations such if you are hungry eat or else you can do other things another example will be one of traffic- IF the light is RED stop the traffic ELSE let the traffic flow.

In Programming this method is very useful because often we need to make sure a certain condition is met before we can continue on other tasks. Below some of the examples of selection:

The first example if the most commonly used if statement. Now if statements are used everywhere in programming and its most common use of selection you can find.

`bool condition = true;`
 	
`if (condition)`
`{`
`Console.WriteLine("The variable is set to true.");`
`}`
`else`
`{`
`Console.WriteLine("The variable is set to false.");`
`}`
	`// Try with m = 12 and then with m = 8.`
	
	`if (m > 10)`
	`{`
	`if (n > 20)`
	`{`
	`Console.WriteLine("Result1");`
	`}`
	`else`
	`{`
	`Console.WriteLine("Result2");`
	`}`
	`}`

In the statement above we are looking for is a condition is then we are going run a certain instruction. If it's not then we run the else instruction. This	In the statement above we are checking if M is greater than 10 and also we are nesting another if statement inside checking is n than 20. It's possible to have nested If statements to serve a purpose.

`int caseSwitch = 1;`
`switch (caseSwitch)`
`{`
`case 1:`
`Console.WriteLine("Case 1");`
`break;`
`case 2:`
`Console.WriteLine("Case 2");`
`break;`
`default:`
`Console.WriteLine("Default case");`
`break;`
`}`
	`int a = 1;`
	`int b = 3;`
	
	`// Use negated expression.`
	`if (!(a == 1 && b == 2))`
	`{`
	`Console.WriteLine(true);`
	`}`
	`// Use binary or version.`
	`if (a != 1 || b != 2)`
	`{`
	`Console.WriteLine(true);`
	`}`

Above example of selection is the one not so often looked at SWITCH statement. It works similarly to the if statement but it has few of its own styles where to checks whether the condition is met. A switch statement will consist of CASES if they match then we run the instruction inside it. Its possible to have 10 20 30 cases if necessary.	Above example is checking the negative values of an if statement. for example when we put a ! symbol in front of a condition its seen as NOT this for example !outside means the outside Boolean will be false in order for this if statement to work. So the example above is stating if A equals 1 AND B equals 2 is not true then we write then we write the line TRUE.

	Second if is checking IF A is NOT equal to 1 OR B is NOT equal to 2 then we write the line TRUE.

Iteration

Iteration is another name of computer loops, which will go through a group of instructions until the end if met. There are infinite loops but they don't serve a specific purpose and often cause the application to crash. So always remember to input an end of the loop to break it or it will cause the application to crash.

There is a old saying in computer science which states if you are writing a line twice you should use a loop. Trying not to repeat the same code over and over again you can almost always use a loop to achieve the same result but efficiently.

There are three main types of loops in computer programming 1- While Loop 2- Do while loop and finally 3 For loop.

int number = 0; **while(number < 5)** **{** **Console.WriteLine(number);** **number = number + 1;** **}**	**do** **{** **Console.WriteLine(number);** **number = number + 1;** **} while(number < 5);**
While Loop - **The while loop simply executes a block of code as long as the condition you give it is true.**	**Do While Loop -** **The opposite is true for the do loop, which works like the while loop in other aspects through. The do loop evaluates the condition after the loop has executed, which makes sure that the code block is always executed at least once.**
int number = 5; **for(int i = 0; i < number; i++)** **{** **Console.WriteLine(i);** **}**	**ArrayList list = new ArrayList();** **list.Add("John Doe");** **list.Add("Jane Doe");** **list.Add("Someone Else");** **foreach(string name in list)** **{** **Console.WriteLine(name);** **}**
For Loop - **The for loop is a bit different. It's preferred when you know how many iterations you want, either because you know the exact amount of iterations, or because you have a variable containing the amount**	**For Each Loop -** **It operates on collections of items, for instance arrays or other built-in list types.** **In the example above its seeking the name string in the list array and will display the strings from that array as it finds them. This is useful when you have a list of objects or arrays and want to find a specific information in bulk.**

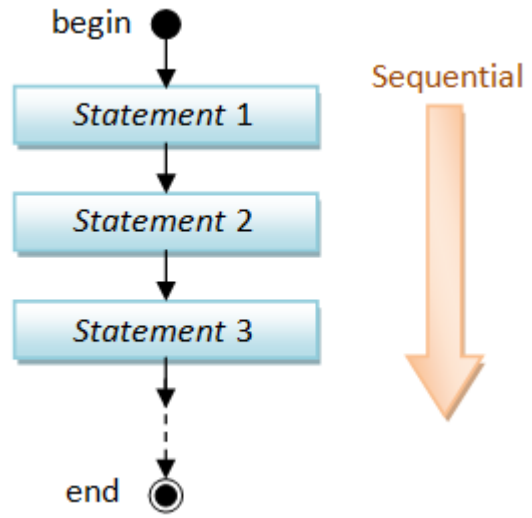
Sequence

There is a set of instructions that the program must follow or execute by first to last order. The order the program executes those instructions are called sequence. Think of this as you going up the stairs you must follow the stair order to get there efficiently or you will go back and forth and end up nowhere. Computer programs are logic driven so they need clear and precise instructions which allow them to start and complete a task. If we give them unclear instruction then they will follow that only to be slowed down but not of their own fault because of our miscalculations.

So its always wise to think of the most logical order which we want out programs to run and then write the code, this way we will encounter less errors and may be even save time in the future.

In a sequence structure, an action, or event, leads to the next ordered action in a predetermined order. The sequence can contain any number of actions, but no actions can be skipped in the sequence. The program, when run, must perform each action in order with no possibility of skipping an action or branching off to another action. [http://www.webopedia.com/TERM/S/sequence.html]

Example




```
int numA = 5;
int numB = 6;

Console.WriteLine("Total is" + (numA+numB);
```

This code reads from top down, same in every programming language. Unless told so in the code it will read top down all the way.

1 integer called **numA** has a value of 5
2 integer called **numB** has a value of 6
3 console application is called to write the line "Total is" which adds **numA** and **numB** are added together.

In this simple example any of the lines cannot do without the other, hence the sequence of code.

Function

function are a self contained modules of code that is designed to achieve a single task, there are multiple types of functions. In programming functions are essential because they are used to encapsulate several smaller tasks into a full module. A function is set to run only once by default however if we need to we make a function run several times but that needs to be stated in the code.

In C# this is a simple function below -

```
public void FunctionNAME()
{
Console.WriteLine("Hi Im inside this function ");
}
```

From top, This function starts with the keyword, PUBLIC which means this function is accessible throughout this program and other classes that are connecting to it.

Second keyword is VOID this keyword is used when the function doesn't have a data type since its not returning a value to the program.

Third keyword is FUNCTIONNAME this can be anything, make sure there is no space in the name. You can call it anything you want but make sure its related to the project itself. Good programming practice is to make call the function in a related name so you can remember and it makes sense, don't call it a1024 because it will only add confusion to the project.

After the function there are double brackets () these are empty because we are not passing any value from there.

After that we have the curly brackets { } they are open and close point of the function. These are very important and one my common mistakes programmers make are they either forget one of them. In visual studio its usually auto completed for the coders but make sure not accidentally delete one of them.

Inside the Curly brackets we put the functions instructions there is where we can do several different things, in this case we are using the console write function to show a line on the screen.

This example below is where a function is passing a value and returning a value back to the program

```
public int Addition(int number1, int number2)

{

int result = number1 + number2;
return result;

}
```

This function starts the same way as before we have a given this a public statement.

Instead of VOID it has a int (integer) type which means this function is required to return a number to the original program.

We are calling this function ADDITION

Inside the brackets we have a int called number1 and the second int called number 2 with a comma as their separator. We can include more in here but we don't need to at the moment.

Inside the function between the curly brackets we started to give this function some instructions to carry out.

First we are declaring a LOCAL variable which means this simple variable called result can be only used inside this ADDITION function. It doesn't exist outside of it.

Result integer will add the two integers number1 and number2 together.

Lastly and importantly this function will RETURN the added value which is stored inside the result integer to the function.

Identify the difference between the two.

In a function we can pass any value such as string, integers, float, picture box and objects. They are all subject to the purpose of the program.

In this tutorial we have discussed selection, iteration, sequence and function.