Mr. Bellavia

LESSON 01 – SEQUENCE & SELECTION (IF-STATEMENT)

In this lesson we will introduce the concept of selection, whereby your programs can start making decisions. Specifically, we will use the if-statement to have our programs make decisions based on input given by the user.

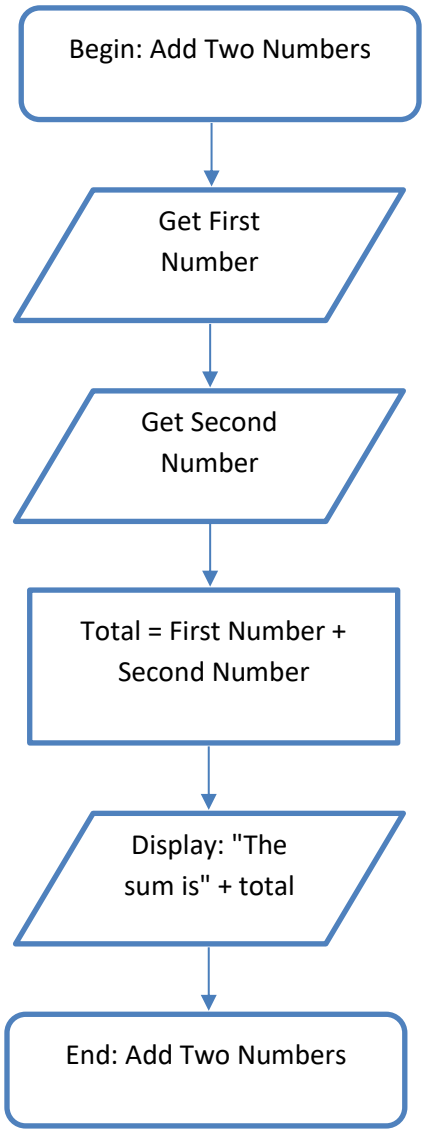
Sections:

Mr. Bellavia

I. SEQUENCE:

In a computer program or an **algorithm** (a set of instructions designed to perform a specific task – *Google*), a **sequence** involves simple steps which are to be executed one after the other. The steps are executed in the same order in which they are written.[1]

We can demonstrate this using a **flowchart** (a step-by-step diagram for mapping out complex situations):



So far, all our programs have followed a similar **linear** dynamic. We can take input, manipulate & perform calculations on it, and output a result. We obviously want our programs to do more, which brings us to out next section: **selection.**

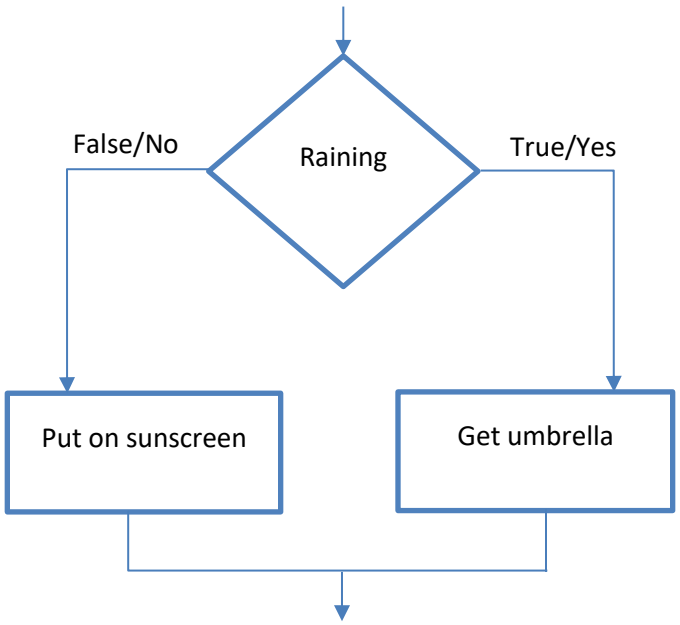
[1] https://courses.p2pu.org/en/courses/2544/content/5290/

II. SELECTION:

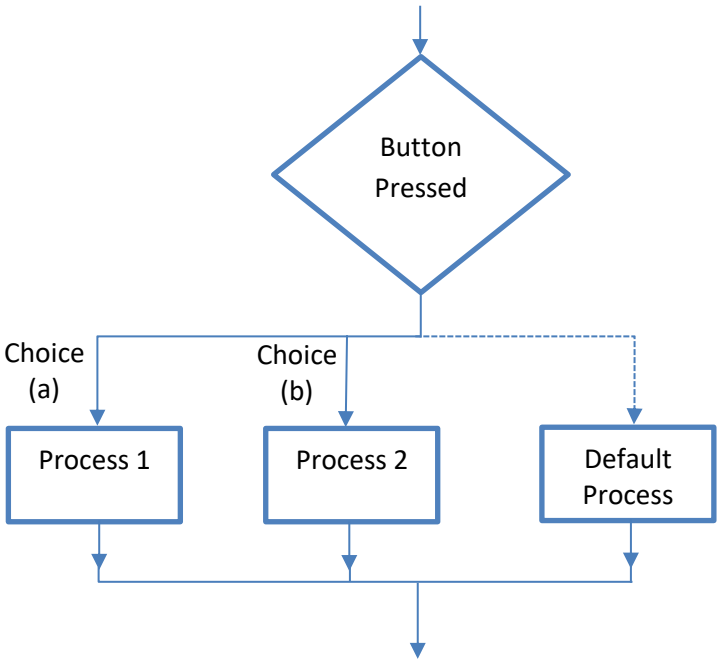
All the programs you use on your computer will need to make **decisions** at one point. For example, you may have a program that asks the user their course average and then your program may need to **decide** what grade letter they should be assigned.

When your program needs to 'make a **decision'**, it is making a **'selection'**. **Selection** is used in a computer program or algorithm to determine which step, or set of steps, is to be executed next. There are two types of selection**: binary selection** (two possible pathways) and **multi-way selection** (many possible pathways).[2] The following flowchart diagrams demonstrate both:

Binary Selection:



Multi-way Selection:



[2] https://courses.p2pu.org/en/courses/2544/content/5290/

III. THE 'IF-STATEMENT' (BOOLEAN LOGIC):

Boolean logic evaluates expressions to **true** or **false** so that your programs can **'make a decision'.** This decision process uses **relational operators** which are used to compare **two** values/variables. The following table lists all the **relational operators:**

Relational Operator	Description
==	Equals
!=	Not equals
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Let's demonstrate how this works with an example:

```
if (1 < 2)
    Console.WriteLine("1 is definitely less than 2!");
```

Output will be:

```
1 is definitely less than 2!
```

This example uses an **'if-statement'** (a very basic one). It basically says: <u>if 1 is less than 2 then output the text *'1 is definitely less than 2!'* to the terminal</u>. This is also known as a **conditional statement:**

In computer science, conditional statements, conditional expressions and conditional constructs are features of a programming language, which perform different computations or actions depending on whether a programmer-specified boolean condition evaluates to true or false. (Wikipedia)

Let's break down this **if-statement:**

```
if (1 < 2)
    Console.WriteLine("1 is definitely less than 2!");
```

The **if-statement** starts with the key word **'if',** followed by an **expression** in brackets. Whatever goes inside the **brackets ( )** of our if-statement is considered our **condition.** In this case our **condition** is **'1 < 2',** which is interpreted as: **if 1 is less than 2.** These conditions will equate to either **'true'** or **'false'.** In this case, it obviously equates to **'true'** since **1** is less than **2.** If the condition equates to **'true'** then the line of code following the if-statement will **execute.** If the condition equates to **'false'** then the line of code following the if-statement will **not execute.**

Let's look at a more practical example:

```
int x = 0;
Console.Write("Please enter a number: ");
x = Convert.ToInt32(Console.ReadLine());

if (x > 10)
    Console.WriteLine("\nYour number is greater than 10!");
```

Sample **Input** & **Output**:

```
Please enter a number: 11

Your number is greater than 10!
```

This program is a little more dynamic than what we have done so far. In this example, we ask the user for a number and the program **checks (determines)** to see if they entered a number **greater than 10 (condition)**. If they did, then we can output an appropriate message **(selection)**.

IMPORTANT: Make sure you **TAB** the line of code over that follows right after the **if-statement.** This is not necessary, but it makes reading your code much easier! Your code will run the exact same with or without a TAB.

IV. BINARY (TWO-WAY) SELECTION ('IF-ELSE'):

Let's look at the code from the previous section:

```
int x = 0;
Console.Write("Please enter a number: ");
x = Convert.ToInt32(Console.ReadLine());

if (x > 10)
    Console.WriteLine("\nYour number is greater than 10!");
```

What if the user in the above program did not enter a number greater than 10? The program would simply continue with no message to the terminal. It would be nice to still have a message outputted to the console. Hence, we have the **'if-else-statement':**

```
int x = 0;
Console.Write("Please enter a number: ");
x = Convert.ToInt32(Console.ReadLine());

if (x > 10)
    Console.WriteLine("\nYour number is greater than 10!");
else
    Console.WriteLine("\nYour number is NOT greater than 10!");
```

Sample **Input** & **Output**:

```
Please enter a number: 8

Your number is NOT greater than 10!
```

In the above code our program now has two choices: It will either print a message saying that the number is **greater than 10, <u>or</u>** it will print a message that the number is **not greater than 10.** It will never print both! This is done with the keyword **'else'**. Our program now has more of a **flow of control:**

In computer science, control flow (or flow of control) is the order in which individual statements, instructions or function calls of an imperative program are executed or evaluated. (Wikipedia)

V. MULTI-WAY SELECTIONS ('IF-ELSE-IF'):

What if we wanted our program to have a third option to choose from? Or even a fourth or fifth option? Let's look at the following example:

```
int x = 0;
Console.Write("Please enter a number: ");
x = Convert.ToInt32(Console.ReadLine());

if (x > 10)
    Console.WriteLine("Your number is greater than 10!");
else if (x < 10)
    Console.WriteLine("Your number is less than 10!");
else
    Console.WriteLine("Your number equals 10!");
```

In the above example, we now have three different paths that the program can take. This is due to the **'else if'** keywords. If you have more than one condition to test/evaluate, then you can use an **if-else-if** statement.

The last **'else'** is usually considered the **'default condition'** and will only execute if all the other **conditions above** are **not true.** Having a default condition is optional. In the above example, if 'x' is not greater or less than 10 then it must be equal to 10 and hence we have our default **'else'** condition.

Note: You can use as many **else-if** statements as you wish. For example:

```
double x = 0;
Console.Write("Enter a temperature (celsius): ");
x = Convert.ToDouble(Console.ReadLine());

if (x <= -10)
    Console.WriteLine("Extermely cold!");
else if (x < 0)
    Console.WriteLine("Below freezing!");
else if (x == 0)
    Console.WriteLine("Freezing!");
else if (x < 10)
    Console.WriteLine("Cold!");
else if (x < 15)
    Console.WriteLine("Cool!");
else if (x < 20)
    Console.WriteLine("Warm!");
else if (x < 25)
    Console.WriteLine("Very warm!");
```

```
else if (x < 30)
    Console.WriteLine("Hot!");
else if (x < 35)
    Console.WriteLine("Very hot!");
else
    Console.WriteLine("Extremely hot!");
```

Notice how important the **order of your if-statements** is for this to work properly! Here is another way we could have written the above code:

```
double x = 0;
Console.Write("Enter a temperature (celsius): ");
x = Convert.ToDouble(Console.ReadLine());

if(x >= 35)
    Console.WriteLine("Extremely hot!");
else if(x > 30)
    Console.WriteLine("Very hot!");
else if(x > 25)
    Console.WriteLine("Hot!");
else if(x > 20)
    Console.WriteLine("Very warm!");
else if(x > 15)
    Console.WriteLine("Warm!");
else if(x > 10)
    Console.WriteLine("Cool!");
else if(x > 0)
    Console.WriteLine("Cold!");
else if (x == 0)
    Console.WriteLine("Freezing!");
else if (x > -10)
    Console.WriteLine("Below Freezing!");
else
    Console.WriteLine("Extermely cold!");
```

IMPORTANT: This is a **single if-statement** block. We are specifically testing the variable 'x,' and **only one** of these statements will execute. Once you put an **'if'** on its own, a new if-statement block is created. For example:

```
if(x >= 35)
    Console.WriteLine("Extremely hot!");
if(x > 30)
    Console.WriteLine("Very hot!");
if(x > 25)
    Console.WriteLine("Hot!");
.
.
.
```

In the above code, there is no longer a single if-statement block. Now we have **multiple if-statements** which all have the potential of being executed separately! This is because we are not using **else-if** statements, but rather just **if** statements. Obviously, this is not what we want in this case!