

LESSON 02 - IF-STATEMENT BRACES & VARIABLE SCOPE



In this lesson we will continue with the concept of selection using the if-statement and look at how we can execute multiple lines of code for a given condition with the use of braces. We will also look at the concept of variable scope throughout our code.

Sections:

- I. [EXECUTING MULTIPLE LINES OF CODE \(BRACES\) PAGE 1](#)
- II. [VARIABLE SCOPE..... PAGE 3](#)
- III. [LOCAL SCOPE VS GLOBAL SCOPE..... PAGE 7](#)

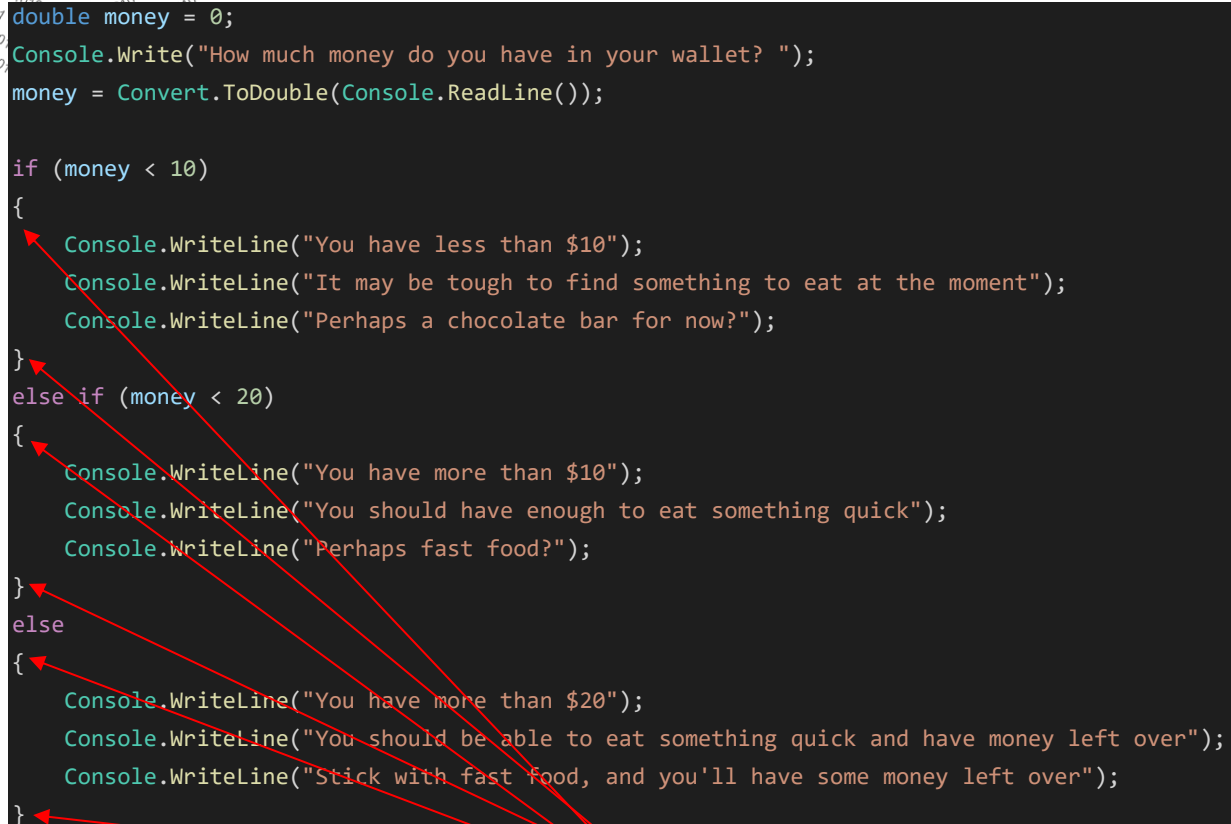
I. EXECUTING MULTIPLE LINES OF CODE (BRACES):

So far, we have seen how to execute a line of code based on a condition using an **if-statement**. For example:

```
int x = 0;
Console.WriteLine("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!");
```

What if there were multiple lines of codes you wanted to execute for a particular condition? We can easily accomplish this with the use of **braces { }**. Let's look at another example:



```
double money = 0;
Console.WriteLine("How much money do you have in your wallet? ");
money = Convert.ToDouble(Console.ReadLine());

if (money < 10)
{
    Console.WriteLine("You have less than $10");
    Console.WriteLine("It may be tough to find something to eat at the moment");
    Console.WriteLine("Perhaps a chocolate bar for now?");
}
else if (money < 20)
{
    Console.WriteLine("You have more than $10");
    Console.WriteLine("You should have enough to eat something quick");
    Console.WriteLine("Perhaps fast food?");
}
else
{
    Console.WriteLine("You have more than $20");
    Console.WriteLine("You should be able to eat something quick and have money left over");
    Console.WriteLine("Stick with fast food, and you'll have some money left over");
}
```

In this example we can see the use of **braces { }** when we want to **execute more than one line for a given condition**. Each condition above has three lines of code that could execute (i.e., each condition has a **block** of code containing 3 lines each). You can have as many lines of code as you wish for any condition. Obviously, each **block** of code will only execute if that given condition is **true**. Here is a breakdown with **pseudo code** (code that is written in plain language for demonstration purposes):

```
if (condition 1 is true)
{
    // block of code to be executed
}
else if (condition 2 is true)
{
    // block of code to be executed
}
. . .
. . .
else
{
    // default block of code to be executed
}
```

II. VARIABLE SCOPE:

As your if-statement construction becomes more complex, it is important to understand variable scope. Before we define variable scope, it is best to start with an example:

```

1  // get number of apples user wishes to buy
2  int numApples = 0;
3  Console.WriteLine("How many apples do you wish to buy ($0.50 each)? ");
4  numApples = Convert.ToInt32(Console.ReadLine());
5  if(numApples <= 0)
6  {
7      Console.WriteLine("Zero apples are free!");
8      Environment.Exit(0);
9  }
10
11 // get the amount of money user has
12 double money = 0;
13 Console.WriteLine("How much money do you have? $");
14 money = Convert.ToDouble(Console.ReadLine());
15 if(money <= 0)
16 {
17     Console.WriteLine("You have no money!");
18     Environment.Exit(0);
19 }
20
21 // run calculations
22 double subTotal = numApples * 0.50;
23 double tax = subTotal * 0.13;
24 double total = subTotal + tax;
25 if(total > money)
26 {
27     Console.WriteLine("Sorry you do not have enough money for the apples!");
28     Environment.Exit(0);
29 }
30 else
31 {
32     Console.WriteLine("\nCalculating...");
33     double moneyLeft = money - total;
34 }
35
36 // output report
37 Console.WriteLine("\nThank you for shopping! Here is your bill:");
38 Console.WriteLine("-----");
39 Console.WriteLine("Number of apples: " + numApples);
40 Console.WriteLine("Cost Per Apple: $0.50");

```

```
41 Console.WriteLine("Sub Total: $" + subTotal.ToString("0.00"));
42 Console.WriteLine("Tax (13%): $" + tax.ToString("0.00"));
43 Console.WriteLine("Total: $" + total.ToString("0.00"));
44 Console.WriteLine("-----");
45 Console.WriteLine("Change: $" + moneyLeft.ToString("0.00"));
```

Before we discuss scope, notice the use of the following statement on **lines 8, 18 & 28**:

```
Environment.Exit(0);
```

This is a C# command that will **exit your program!** In the example above, we are **exiting** the program if the user enters less than or equal to 0 apples (**line 8**), less than or equal to 0 for money (**line 18**), or if the user does not have enough money for the purchase (**line 28**).

If you try to run the above program, you will get a **build error**. This is because on **line 45**, the variable '**moneyLeft**' does not exist in the current context. Why? Because of its scope. If you notice, the '**moneyLeft**' variable is declared on **line 33**, but it exists inside the **braces { }** of the **else block**. That means that that variable **can only be used inside those braces**. This is an example of **variable scope**:

In computer programming, the scope of a name binding—an association of a name to an entity, such as a variable—is the part of a program where the name binding is valid, that is where the name can be used to refer to the entity (Wikipedia).

To fix this problem, we should declare the variable outside of this if-statement, specifically before this if-statement. So, starting from **line 21** we can do the following:

```
21 // run calculations
22 double subTotal = numApples * 0.50;
23 double tax = subTotal * 0.13;
24 double total = subTotal + tax;
25 double moneyLeft = 0;
26 if(total > money)
27 {
28     Console.WriteLine("Sorry you do not have enough money for the apples!");
29     Environment.Exit(0);
30 }
31 else
32 {
33     Console.WriteLine("\nCalculating...");
34     moneyLeft = money - total;
35 }
36
37 // output report
38 Console.WriteLine("\nThank you for shopping! Here is your bill:");
39 Console.WriteLine("-----");
40 Console.WriteLine("Number of apples: " + numApples);
41 Console.WriteLine("Cost Per Apple: $0.50");
42 Console.WriteLine("Sub Total: $" + subTotal.ToString("0.00"));
43 Console.WriteLine("Tax (13%): $" + tax.ToString("0.00"));
44 Console.WriteLine("Total: $" + total.ToString("0.00"));
45 Console.WriteLine("-----");
46 Console.WriteLine("Change: $" + moneyLeft.ToString("0.00"));
```

Now we have moved our declaration of our variable **'moneyLeft'** to **before** the if-statement, and then used our variable both **inside** our if-statement and **after** our if-statement. Our program will work fine now since our variable **'moneyLeft'** is now visible to all our code (i.e., not just from within the braces).

Sample Input & Output:

How many apples do you wish to buy (\$0.50 each)? 5
How much money do you have? \$20

Calculating...

Thank you for shopping! Here is your bill:

Number of apples: 5
Cost Per Apple: \$0.50
Sub Total: \$2.50
Tax (13%): \$0.33
Total: \$2.83

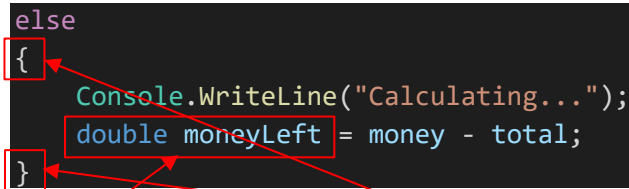
Change: \$17.18

III. LOCAL SCOPE VS GLOBAL SCOPE:

What was demonstrated in the last section was **variable scope**. To identify further, we saw two types of scope: **local scope** and **global scope**.

- **Local Scope:**

When we had initially declared our variable inside the **braces** of our **else condition** in the previous section, that variable had a **local scope** within the braces only, which means that that variable was only visible from where it was declared and **within the braces only**:

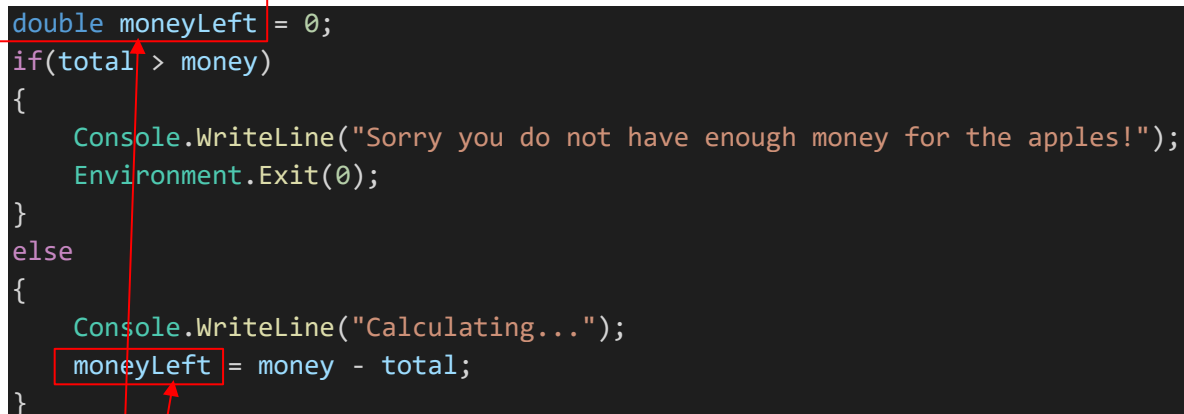


```
else
{
    Console.WriteLine("Calculating...");
    double moneyLeft = money - total;
}
```

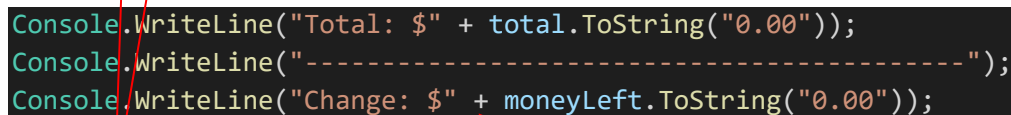
Variable is only visible within the braces of this **else statement**.

- **Global Scope:**

When we moved the declaration to the top of our if-statement in the previous section, that variable had a **global scope** for the entire program, which means that that variable was visible from where it was declared to the end of our program:



```
double moneyLeft = 0;
if(total > money)
{
    Console.WriteLine("Sorry you do not have enough money for the apples!");
    Environment.Exit(0);
}
else
{
    Console.WriteLine("Calculating...");
    moneyLeft = money - total;
}
```



```
Console.WriteLine("Total: $" + total.ToString("0.00"));
Console.WriteLine("-----");
Console.WriteLine("Change: $" + moneyLeft.ToString("0.00"));
```

Variable is visible to entire program.