Introduction to Programming Unit
Lesson 3 Logic: If and Else Statements
Teacher's Guide

Hello and welcome back to the Teachers Edition of a Guided Intro to Programming. This guide will mirror the exact same guide that the students will be using except this guide will have tips and advice on how to help students if they encounter errors.

Now that you have learned about the simple math that we will be using in the future, and you have learned about the print and scan functions we can now start moving into logic. The best place to start learning about logic is with the *if* and *else* statements. On top of learning these statements we will also be learning a few simple logic operations which include: is equal, not equal, less/greater than, less/greater than or equal to, "or" statements, and "and" statements. After this lesson you will have a basic knowledge of how logic in a computer works, as well as being able to do simple logic operations to evaluate inputs.

Let's start by learning these new logic operations as they will be important to creating the logical *if* and *else* statements. The logic operations we will be using today, how to use it, and what it does is shown below.

Logic Operations	How to use it	What it does
Equals ==	X==Y	Instead of making X equal to Y this arithmetic detects if X is equal to Y
Not Equal !=	X!=Y	Detects if X is not equal to Y
Less Than <	X<Y	Detects if X is less than Y

Greater Than >	X>Y	Detects if X is greater than Y
Less Than or Equal to <=	X<=Y	Detects if X is less than or equal to Y
Greater Than or Equal to >=	X>=Y	Detects if X is greater than or equal to Y
Modulus %	X%Y	Modulus is a math operation that is usually used in logic. It divides X by Y and gives you a remainder

As a note to tell students, modulus is a math operation as stated above that will give you the remainder of that division. If X is divisible by Y then the modulus will return 0, otherwise it will return the remainder. For example, 8%3 = 2 because the closest number 3 can get to 8 without going over is 6, 8-6 = 2.

Before we move on to discussing the "or" and "and" statements we need to discuss what logic in computing is. Logic in computing is in a simple sense a series of true or false statements. Using the arithmetic above we can make true and false statements with what we have seen. For example, X==Y, if X is equal to Y then the statement returns true, and if X is not equal to Y then the statements returns false. The truth and falsity of a statement determines what the computer does next in the programs you write. This will be shown when we get into the *if* and *else* statements.

What we saw above is a simplified version of how logic works in computing. In a more mechanical sense logic still does work on a true/false system in computers but we know this as binary. Binary is the lowest level language that a computer uses and is what we call machine code. This is because a computer is a series of logic processes that result in a 1 or a 0, where 1 is true and 0 is false. It is important to know this for the future if you plan to do anything with computers and is good to know but not necessary for what we will now discuss.

Now that we know about logic and binary, we can discuss the "or" and "and" statements. The "or" and "and" statements take the input of two of the logic operations above. They can be two different operations or the same one just with different parameters. In simple terms the "or" statement will take both logic operations and uses them to determine its output. We see this in a true false table to visualize the output of the statement.

Logic Operation 1	Logic Operation 2	Result
True	True	True
True	False	True
False	True	True
False	False	False

The students should realize that in order to make a statement true in the "or" logic function only one of the two or however many logic operations there needs to be true.

Like the "or" statement, the "and" statement also compares two logic operations to return either true or false. However, in the "and" statement both operations must be true to return true, otherwise the whole statement is false. See the true false table below:

As an assignment to get students ready for logic, have them try to remember the truth tables for "or" and "and" logic functions. This will help them later down the line and is just good practice to understand basic logic.

Logic Operation 1	Logic Operation 2	Result
True	True	True
True	False	False
False	True	False
False	False	False

The students should realize that in order to make a statement true in the "and" logic function every logic operation there needs to be true. Otherwise, it is considered false.

If you are having a hard time understanding this, do not worry. Logic may be a new concept to many of you and that is okay. We all need to start somewhere and we cannot be expected to understand every new concept as we are given it. I will show some examples at the end to help you better understand how this works. But before I do that, I need to show you the *if* and *else* statements.

The *if* and *else* statements are what we refer to in computing as conditional statements, meaning that a condition needs to be met in order to do what is in the statement. For example, let's say

you want to buy a new video game that cost $60 and you only have $50 in your bank account, we can write a simple *if* and *else* statement using this and it would look like this:

If Bank Account is equal to $60 then buy the game, else save money.
The *if* statement takes a condition you give it and if that condition is met then it will do what is in the statement, and if the condition is not met it either stops or can go to an *else* statement where it will run the function in the *else* statement.
Now let's see what this looks like in code:
If(condition){
Function;
}else{
Function;}
This is the proper way to write an *if* and *else* statement in *C*. So, let me show you the example I made above in code so you may better understand the concept.

main.c

```c
#include <stdio.h>

int main(void) {

  int Account = 0;

  printf("You want to buy a new game that costs $60.\n");
  printf("How much money is in your bank account: $");
  scanf("%d", &Account);

  if(Account == 60){
    printf("You can afford the game, have fun playing.\n");
  }else{
    printf("Save some more money so you can buy the game.\n");
  }
  return 0;
}
```

This is the output you get when you have $60

```
clang-7 -pthread -lm -o main main.c
./main
You want to buy a new game that costs $60.
How much money is in your bank account: $60
You can afford the game, have fun playing.
```

And this is the output you get when you don't have $60

```
clang-7 -pthread -lm -o main main.c
./main
You want to buy a new game that costs $60.
How much money is in your bank account: $50
Save some more money so you can buy the game.
```

It is also important to note that you can write just an if statement and not have the else after it. Knowing how to use the *if* and *else* statements is essential in programming. This is because they are used to do almost any high-level program, but for now it is good that you just get some simple experience with it. And with that,

Congratulations

You have completed lesson 3 of a Guided Introduction to programming, come back next time to learn about loops.

Simple programs that the students can do to test their knowledge on what they learned today and can be used as assignments:

1. Finding if a number is even or odd: its as simple as if(x%2==0) which will tell you if X is even.
2. The program I did above where the student needs to meet a condition and whether or not they reached that condition there is an appropriate output.

It is also important that when checking for errors in the student's code that you analyze these few things first:

1. Check any errors that could occur outside of the if and else statements that may have been discussed in previous lessons

2. Check their logic operation to see if it is written correctly and that it is in the parentheses.
3. Check their function in the if/else statements and be sure they contain no errors and that they are written in between the "{" Braces "}" like shown in the code above.

Glossary

Binary: a binary number is a number expressed in the base-2 numeral system or binary numeral system, which uses only two symbols: typically, "0" and "1"

Conditional Statements: Features of a programming language, which perform different computations or actions depending on if condition evaluates to true or false.

Logic: A system or set of principles underlying the arrangements of elements in a computer or electronic device so as to perform a specified task.

Machine Code: A computer programming language consisting of binary instructions which a computer can respond to directly.

Parameter: any characteristic that can help in defining or classifying a particular system

Assessment

1. With the following information, write a conditional statement:
You want to drive to Disney, and you are 50 miles away, but you only have 20 miles worth of gas, can you make it without stopping?

2. Reading the definition of the modulus function, what is 15%4?

3. What is 15%5?

Extensions and/or Additional Resources

- Link to Lesson 3 Video
- Link to Programming Vocabulary
- https://www.w3schools.com/
- https://www.geeksforgeeks.org/c-programming-language/
- https://replit.com/languages/c

EDUCATIONAL OUTREACH
FLORIDA POLYTECHNIC UNIVERSITY

Created By:
Ryan Floyd, Computer Science, ‘21

Thank you for downloading this lesson, please take a moment to complete our survey