Flow Control

What is flow control?

- By using key words, flow control alters the flow of a program
- These conditionals are based on a Boolean expression (true or false)
- To evalute expressions into Boolean values, we use comparison operators

Comparison Operators

Operator	Description
==	equal value
!=	unequal value
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

- Comparison operators are used to compare two values to each other
  - A comparison expression will always result in a boolean

```
int x = 5 + 7;
bool test1 = x == 12;    //true
bool test2 = x < 6;      //false
bool test3 = x != 99;    //true
bool test4 = x >= 12;    //true
```

If Statement

- The if keyword is used to check if an expression is true. If it is true, a statement is executed.

"If {expression} is TRUE then do these instructions"

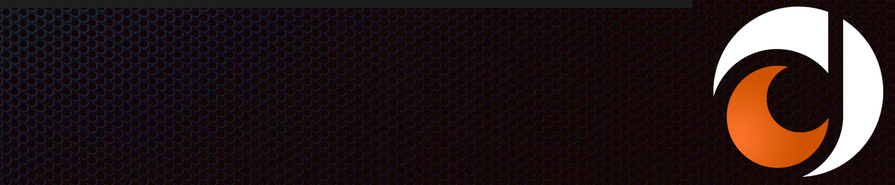
What an if statement looks like:

```
double sellPriceDollars = 7.50;
if(sellPriceDollars > 5.50)
{
    Console.WriteLine("Made a profit on the sale!");
}
```

What an If / Else Statement looks like

```
double sellPriceDollars = 7.50;

if(sellPriceDollars > 5.50)
{
    Console.WriteLine("We made a profit on the sale");
}
else
{
    Console.WriteLine("We didn't make a profit");
}
```



What an if-else if-else statement looks like:

```
double sellPriceDollars = 7.50;
if(sellPriceDollars == 5.50)
{
    Console.WriteLine("Broke even on the sale");
}
else if(sellPriceDollars >= 5.51)
{
    Console.WriteLine("Made a profit on the sale!");
}
else
{
    Console.WriteLine("Lost money on the sale");
}
```

If, Else If, Else Statement

- An if/else statement:
  - ALWAYS starts with an ‘if’ clause
  - May contain any number of ‘else if’ clauses
  - May optionally end with and ‘else’ clause
    - An ‘else’ clause cannot have a condition
    - An ‘else’ clause will ALWAYS represent the end of the ‘if/else’

Logical Operators

-Logical operators are used to join two or more comparison expressions together.

-&& (AND)
  -Requires all individual expressions to be true.
  -One false will make the entire expression false.
- || (OR)
  -Requires only one of the individual expressions to be true.
  -For the entire entire expression to be false, all individual expressions must be false.
-! (NOT)
  -Literally means "opposite of"

Logical Operators (&&)

-If ALL individual expressions are true, the entire expression is true

```
int age = 16;

bool isTeenager = age >= 13 && age <= 19;
      //true
```

-If ANY of the individual expressions are false, the entire expression is false

```
int age = 17;
bool hasDriversLicense = false;
bool canDrive = age >= 16 && hasDriversLicense;
    //false
```

Logical Operators (||)

-If ALL individual expressions are true, the entire expression is true

```
string name = "Peter Parker";
bool isSpiderman = name == "Peter Parker" || name == "Miles Morales";
    //true
```

-If ALL of the individual expressions are false, the entire expression is false

```
int cash = 20;
bool hasCreditCard = false;
int price = 25;
bool canPurchase = cash >= price || hasCreditCard;
    //false
```

Logical Operators

- We can also use both logical operators in the same expression to create more complicated and restrictive operations

```
int age = 15;
bool hasLearnersPermit = true;
bool isWithParents = true;
bool canDrive = age >= 16 || (hasLearnersPermit && isWithParents && age >= 15);
//true
```

Logical Operators (!)

- Returns the opposite value

```
bool isStudent = false;
Console.WriteLine(isStudent);   //false
isStudent = !isStudent;
Console.WriteLine(isStudent);   //true
Console.WriteLine(!isStudent);  //false (value is still true)
```

Switch Statement

- A switch statement tests a value (switch) from the variable or expression against a list of values (cases).
- If the switch value matches a case value, the code following the case value is executed.
- The default case is executed if none of the other cases are met.

What a switch statement looks like:

```
public void WatchMovie()
{
    Console.Write("Please choose a movie to watch \n");
    string movie = Console.ReadLine();

    switch (movie)
    {
        case "Toy Story":
            Console.WriteLine("Enjoy " + movie);
            break;
        case "Saving Private Ryan":
            Console.WriteLine(movie + " is a great war film");
            break;
        case "Star Wars The Force Awakens":
            Console.WriteLine("Enjoy the new " + movie + " film!");
            break;
        default:
            Console.WriteLine("Not a valid movie to watch");
            break;
    }
}
```

Difference Between If and Switch

- If statement is used when there is a Boolean expression (a true/false selection). Essentially, a series of Boolean checks.
- A switch-case is used when a specific value is desired.
- If you see yourself using an if statement to check if values are equal to each other, then it makes more sense to use a switch-case

Break

- To terminate a block of code in a switch statement, while loop, or for loop, there is the break keyword.
- The break statement can be used to break out of an endless loop.
- In the case of a switch, it can be used to break out of the switch-case when a value is satisfied.
- The break keyword will exit a loop or code block completely.

What the break keyword looks like:

```
public void WatchMovie()
{
    Console.Write("Please choose a movie to watch \n");
    string movie = Console.ReadLine();

    switch (movie)
    {
        case "Toy Story":
            Console.WriteLine("Enjoy " + movie);
            break;
        case "Saving Private Ryan":
            Console.WriteLine(movie + " is a great war film");
            break;
        case "Star Wars The Force Awakens":
            Console.WriteLine("Enjoy the new " + movie + " film!");
            break;
        default:
            Console.WriteLine("Not a valid movie to watch");
            break;
    }
}
```

Scope

-Scope is how we describe the "visibility of variables.

-In many languages, including C#, scope is often defined by curly brackets.

```
string instructor = "Nevin";
if(instructor == "Nevin")
{
    string phrase = "You lucky students!";
    Console.WriteLine(phrase);
}

Console.WriteLine(phrase);
```

The name 'phrase' does not exist in the current context

Show potential fixes (Alt+Enter or Ctrl+.)

Key Takeaways:

- Comparison operators compare two things and return a boolean.
  - (==, !=, >, >=, <, <=)
- Logical operators allow us to group together logical statements.
  - (&&, ||, !)
- if/else statements are a form of flow control.
- switch/case statements are also a form of flow control.
- Scope describes where a variable *exists.*
  - A variable's scope is determined by where that variable is *declared*.

Review Questions

- What is the difference between an if statement and switch-case?
- Does an if statement need to include ‘else’ in order to be valid?
- Does a switch-case need to have a ‘default’ in order to be valid?
- When is it appropriate to use the break keyword?