CONTROL STATEMENTS AND LOOPS

BOOLEAN EXPRESSIONS

BOOLEAN EXPRESSIONS AND CONTROL STATEMENTS

Recall that a boolean expression is one that evaluates to either true or false. We can use these expressions to determine the flow of our control statements.

RELATIONAL OPERATORS

The six relational operators (listed in following slide) are used to compare operands which are primitive data types.

Operands may be literals, variables, arithmetic expressions, or keywords.

RELATIONAL OPERATORS

Operator	Name
==	Equality
!=	Inequality
>	Greater Than
<	Less Than
>=	Greater Than or Equal
<=	Less Than or Equal

EXAMPLES ON BOOLEAN EXPRESSIONS

```
discountPercent == 2.3 // equal to a numeric literal
letter == 'y' // equal to a char literal
isValid == false // equal to the false value
subtotal != 0 // not equal to a numeric literal
years > 0 // greater than a numeric literal
i < months // less than a variable
subtotal >= 500 // greater than or equal to a numeric literal
quantity <= reorderPoint // less than or equal to a variable
isValid // isValid is equal to true
!isValid // isValid is equal to false
```

LOGICAL OPERATORS

&& And

| | Or

! Not

SELECTION STATEMENTS

- If Statement
- Switch Statement

THE IF STATEMENT

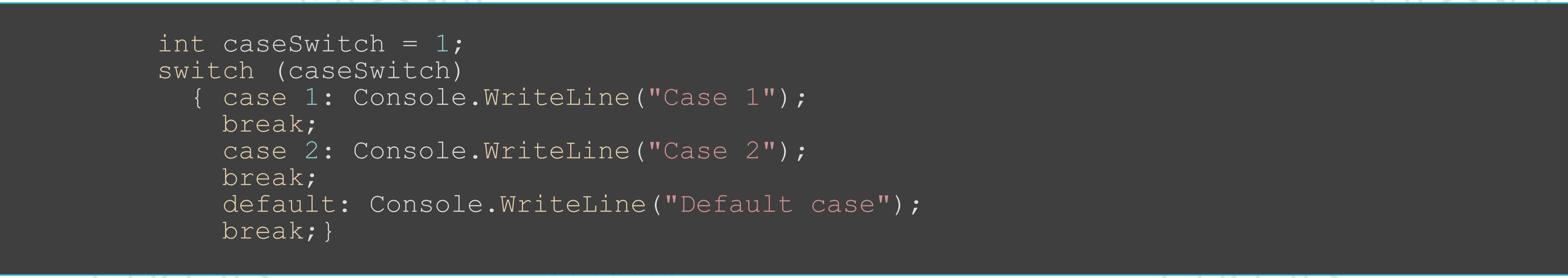
An if statement identifies which statement to run based on the value of a Boolean expression. Example:

```
bool condition = true;
if (condition)
       {
           Console.WriteLine("Variable is set to true.");
       }
else
       {
           Console.WriteLine("Variable is set to false.");
       }
```

THE SWITCH STATEMENT

The switch statement is a control statement that selects a switch section to execute from a list of candidates.

Example:



```
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;}
```

LOOPS

- The while loop
- The do loop
- The for loop
- The for each loop

THE WHILE LOOP

A while loop statement repeatedly executes a target statement as long as a given condition is true.

```
while(Boolean_expression)
{
        //Statements
}
```

THE DO LOOP

A do loop is similar to a while loop, except that a do loop is guaranteed to execute at least one time.

```
do
{
        //Statements
}while(Boolean_expression);
```

THE FOR LOOP

A for loop allows you to write a loop that needs to execute a specific number of times. A for loop is useful when you know how many times a task is to be repeated.

```
for(initialization; Boolean_expression; update)
{
   //Statements
}
```

THE FOREACH LOOP

This is mainly used to iterate through a collection of elements inside collections, such as arrays.

```
foreach(declaration in expression)
{
       //Statements
}
```

FOREACH EXAMPLE

```
int [] numbers = {10, 20, 30, 40, 50};

foreach(int x in numbers ){
   Console.WriteLine( x );
   Console.WriteLine(",");
}
Console.WriteLine("\n");
String [] names ={"James", "Larry", "Tom", "Lacy"};
foreach( String name in names ) {
   Console.WriteLine( name );
   Console.WriteLine(",");}
```

SIMPLE CONTROL STATEMENTS

COMPARE NUMERIC VARIABLES

We can use the six relational operators to compare numeric variables. These allow us to write Boolean expressions (which evaluate to either true or false).

CONTROL STATEMENTS

BREAK STATEMENT

The break statement is used to exit the current loop. In the case of multiple nested loops, a labeled break statement may be used to differentiate.

CONTROL STATEMENTS

CONTINUE STATEMENT

The continue statement is used to skip any remaining statements in the current loop and jump to the top of the current loop. A labeled continue statement may be used to jump to the top of a labeled loop.

RECAP

WHAT YOU SHOULD KNOW AT THIS POINT:

- How to evaluate Boolean expressions.
- Know how to use logical and relational operators.
- Use if and switch statements
- Use for, while, do while, and enhanced for loop.
- Comparing numeric variables.
- Comparing string variables.
- How to use continue and break statements.