Adding Decision and Iteration Statements in C#

Gill Cleeren
CTO Xpirit Belgium

@gillcleeren | xpirit.com/gill

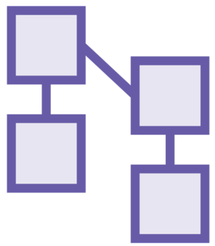
Agenda

- Working with Boolean values
- Making decisions with the if statement
- Using the switch statement
- Adding iterations

Working with Boolean Values

Boolean Values

True or false



bool type (Boolean backing type)

Boolean operators

```
bool c = true;
Console.WriteLine(c);//Writes True to the console
```

Using a Boolean Value

Using Relational Operators

Operator	Example
==	a == b
!=	a != b
> or <	a > 10
>= or <=	a <= 5

```
age == 45 //True if value of age is effectively equal to 45, otherwise false
age != 0  //True if age is not equal to 0
```

Using Logical Operators

Using Boolean Logical Operators: &&

```
bool validAge;

validAge = (age >= 18) && (age <= 65);
```

True if either of the expressions is true, otherwise false

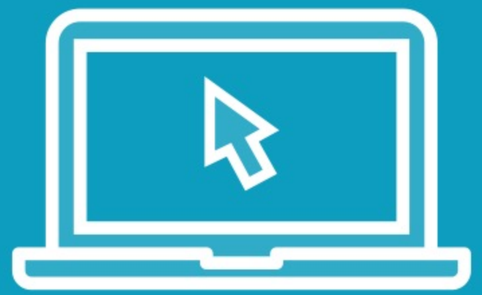
Using Boolean Logical Operators: ||

```
bool validAge;

validAge = (age >= 18) || (age <= 65);
```

True if any of the expressions is true, false only if both are false

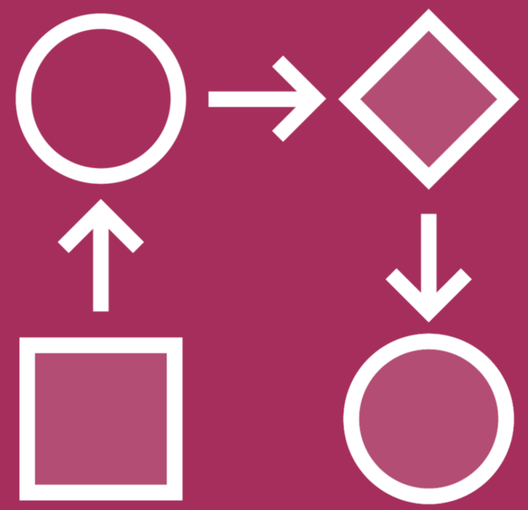
Demo



Working with relational operators

Using Boolean logical operators

Making Decisions with the if Statement



Flow of Execution

Won’t be a straight path
Depends on values
Different logic needs to be executed

A New Requirement

If the person applying for the job is under 18, we can't hire them.

If the person applying is older than 65, we can't hire them.

Structure of an if Statement

```
if(some Boolean expression)
{
    //Other statements
}
else
{
    //Other statements
    //The else block is optional
}
```

Using an if Statement

```
if(age < 18)
{
    Console.WriteLine("Too young to apply");
}
```

Using an if Statement

```
if(age < 18)
{
    Console.WriteLine("Too young to apply");
}
else
{
    Console.WriteLine("Great, you can now start with your application!");
}
```

Using an if Statement

```
if(age < 18)
    Console.WriteLine("Too young to apply");
else
    Console.WriteLine("Great, you can now start with your application!");
```

Using an if Statement

```
if(age < 18)
    Console.WriteLine("Too young to apply");
    Console.WriteLine("Please try again later!");
     //We need curly braces here!
else
    Console.WriteLine("Great, you can continue!");
```

This Won't Work...

```
if(age = 100)
{
    //Send mail
}
```

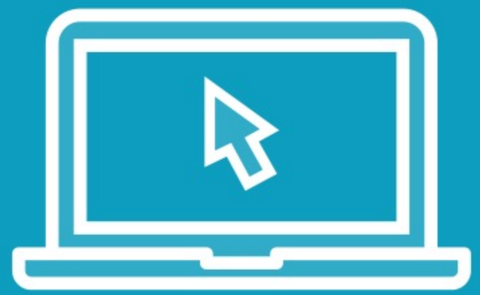
Adding Multiple Conditions

```
if(some Boolean expression)
{
    //Other statements
}
else if (other Boolean expression)
{
    //Other statements
}
...
else
{
    //Other statements
}
```

Using an else if Block

```
if (age < 18)
{
    Console.WriteLine("Too young to apply");
}
else if (age > 65)
{
    Console.WriteLine("Sorry, the selected age is too old");
}
else
{
    Console.WriteLine("Great, you can continue!");
}
```

Demo



Using if statements

Adding multiple conditions

Using the Switch Statement

Too Many Options...

```
if(condition 1)
    ...
else if(condition 2)
    ...
else if(condition 3)
    ...
else if(condition 4)
    ...
else if(condition 5)
    ...
else if(condition 6)
    ...
...
else
    ...
```

Structure of a switch Statement

```
switch(expression)
{
    case constant expression 1:
        //Other statements
        break;
    case relational expression 2:
        //Other statements
        break;
    ...
    default:
        //Other statements
        break;
}
```

Using a switch Statement

```
switch (age)
{
    case < 18:
        Console.WriteLine("Too young to apply");
        break;
    case > 65:
        Console.WriteLine("Sorry, the selected age is too old");
        break;
    case 42:
        Console.WriteLine("Wow, exactly what we are looking for");
        break;
    default:
        Console.WriteLine("Great, you can continue");
        break;
}
```

Using the switch Statement

Works for most data types but not for float and double

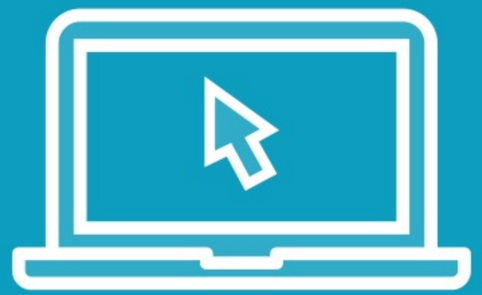
Case labels use a pattern: constant or relational.

Each case must be unique

First “true” will get executed (top to bottom)

Default can be placed wherever we want, always evaluated last

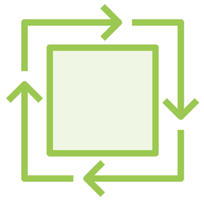
Demo



Using the switch statement

Adding Iterations

The Need for Iterations



Continue executing a task (looping)

Often used in combination with counter

Ask input until stop is reached

Keep reading files from disk

Loop Options in C#

while

do-while

for

Creating a while Loop

```
while (Boolean expression)
{
  //statements
}
```

```
while (Boolean expression)
{
   //statements
}
```

Condition is tested before the loop runs

Statements will get executed as long as expression is true

Braces are required if more than one statement must be executed

We can create infinite loops!

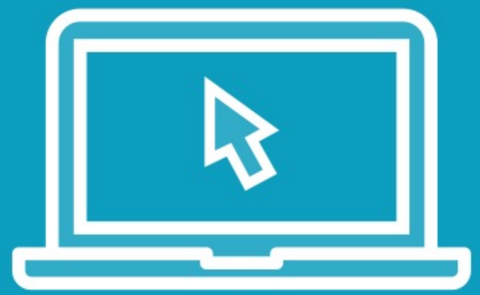
Creating a while Loop

```
int i = 0;

while (i < 10)
{
    Console.WriteLine(i);
    i++;
}
```

```
Output:
 0
 1
 2
 3
 4
 5
 6
 7
 8
 9
```

Demo



Creating a while loop

Creating a nested loop

Creating a do-while Loop

```
do {
  //statements
}
while (Boolean expression);
```

Creating a do-while Loop

```
int i = 0;

do
{
    Console.WriteLine(i);
    i++;

} while (i < 10);
```

```
Output:
 0
 1
 2
 3
 4
 5
 6
 7
 8
 9
```

A do-while Will Always Get Executed at Least Once

```
int i = 10;

do
{
    Console.WriteLine(i);
    i++;

} while (i < 10);
```

```
Output:
 10
```

Creating a for Loop

```
for (initialization; Boolean; iterator)
{
  //statements
}
```

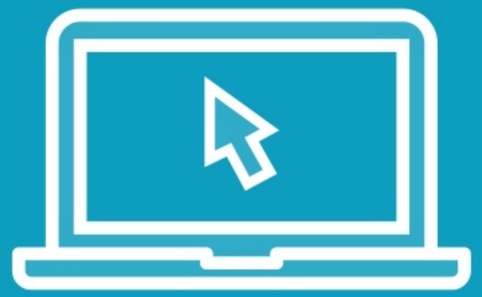
Creating a for Loop

```
int sum = 0;

for (int i = 0; i < 10; i++)
{
    sum = sum + i;
}

Console.WriteLine(sum);
```

Demo



Creating more loops

Adding break and continue

Debugging loops

Summary

Our C# code will need to follow different paths

if and switch statements allow to evaluate values

while, do-while and for create iterations

Up next:

Working with methods