7SENG011W Object Oriented Programming

More Selection Statements, Blocks, Loops

Dr Francesco Tusa

Readings

The topics we will discuss today can be found in the books

- Hands-On Object-Oriented Programming with C#
    - Chapter: Overview Of C# As A Language
- Programming C# 10.0
    - Chapter: Basic Coding In C#
- C# online documentation
    - Boolean operators
    - Selection statements
    - Iteration statements

Outline

- Summary of the previous lecture
- Loops
- More on selection statements: switch-case

Logical operators

&&: logical **AND**

true if both operands are true, *false* otherwise

```
// c = 1, d = 4
bool b = c > 0 && d < 5; // b contains true
```

||: logical **OR**

true if at least one of the operands is true, *false* otherwise

```
bool b = c > 3 || d < 5; // b contains true
```

Logical operators

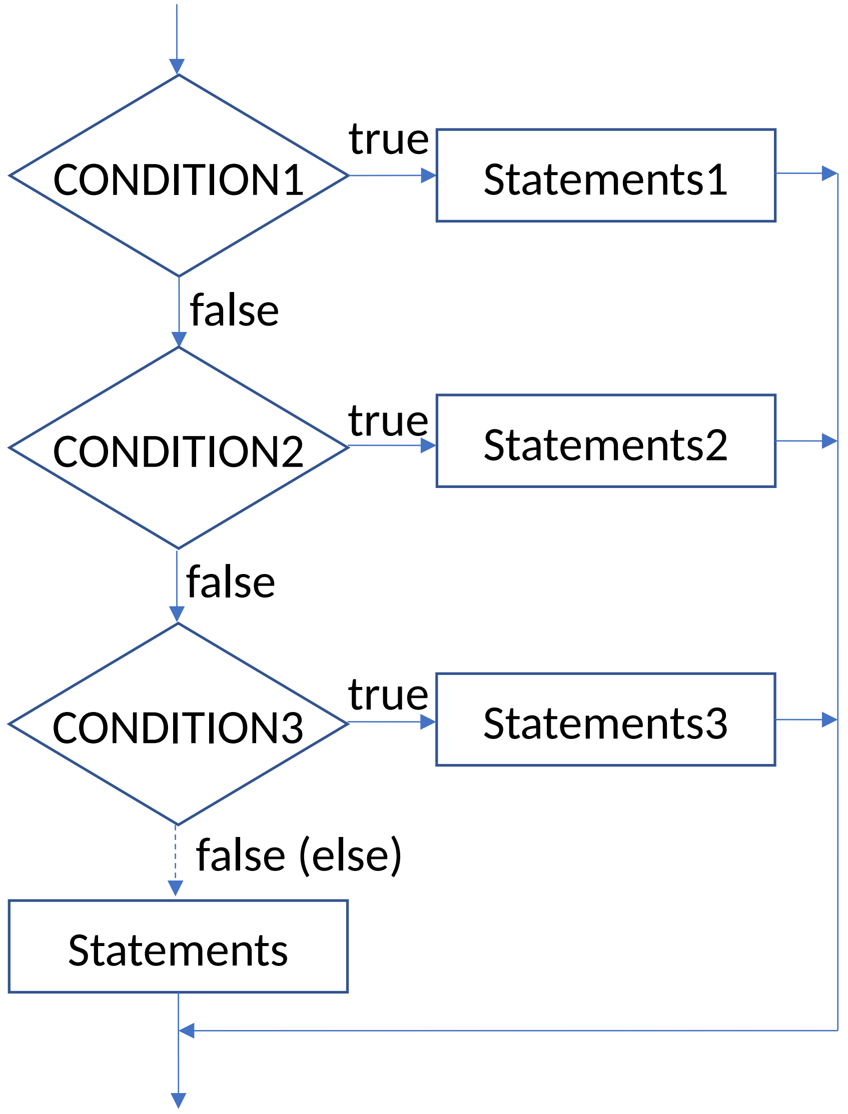
!: logical **NOT**

- *unary* operator that *changes* the value of its operand
- if the operand is *true*, the result is *false*; if the operand is *false*, the result is *true*

```
int c = 2;
int d = 3;
bool flag = c > 0; // true
bool result = ! flag && d < 5; // false
```

Selection statements: **if-else-if**

```
if (CONDITION1) {
        STATEMENTS1
} else if (CONDITION2) {
        STATEMENTS2
} else if (CONDITION3) {
        STATEMENTS3
} else {
        STATEMENTS
}
```



Exercise (Homework)

- A sensor collects temperature measurements *T* (in Celsius)
- Using an appropriate selection statement, write a program that prints different messages on the screen:
  - "Normal" when T <= 24 C
  - "Warning" when 24 C < T <= 30 C
  - "Critical" when T > 30 C
- *T* should be typed in from the keyboard (emulate a sensor value)
  - Hint: use Convert.ToInt32() and Console.ReadLine()

Review of the Homework

```
static void Main(string[] args)
{
    int temperature = Convert.ToInt32(Console.ReadLine()); // instructions have been combined
    if (temperature > 30)
    {
        Console.WriteLine("Critical");
    }
    else if (temperature > 24)
    {
        Console.WriteLine("Warning");
    }
    else
    {
        Console.WriteLine("Normal");
    }
}
```

"Normal": T <= 24 C
"Warning": 24 C < T <= 30 C
"Critical": when T > 30 C

Review of the Homework

```
static void Main(string[] args)
{
    int temperature = Convert.ToInt32(Console.ReadLine());
    if (temperature > 30)
    {
        Console.WriteLine("Critical");
    }
    else if (temperature > 24 && ~~temperature <= 30)~~ Duplicated Test
    {
        Console.WriteLine("Warning");
    }
    else
    {
        Console.WriteLine("Normal");
    }
}
```

"Normal": T <= 24 C
"Warning": 24 C < T <= 30 C
"Critical": when T > 30 C

Nested blocks

```
class Example
{
        static void Main(string[] args)
        { // beginning of Main block

                int x = 10, y = 5; // values are assigned to x and y
                int sum = x + y;

                if (sum < 20)
                { // beginning of nested block
                        Console.WriteLine(sum + " is less than 20");
                } // end of nested block
        } // end of Main block
}
```

The code inside this block is able to access the variable (*sum*) declared in the parent block

Nested blocks: variable not in scope

```
class Example
{
        static void Main(string[] args)
        { // beginning of Main block

                int x = 10, y = 5; // values are assigned to x and y
                int sum = x + y;

                if (sum < 20)
                { // beginning of nested block
                        Console.WriteLine(sum + " is less than 20");
                        int willNotWork = sum * 5;
                } // end of nested block
                Console.WriteLine(" willNotWork is " + willNotWork);
        } // end of Main block
}
```

willNotWork only exists **within** the if block

Outline

- Summary of the previous lecture
- Loops
- More on selection statements: switch-case

Printing numbers (up to 3)

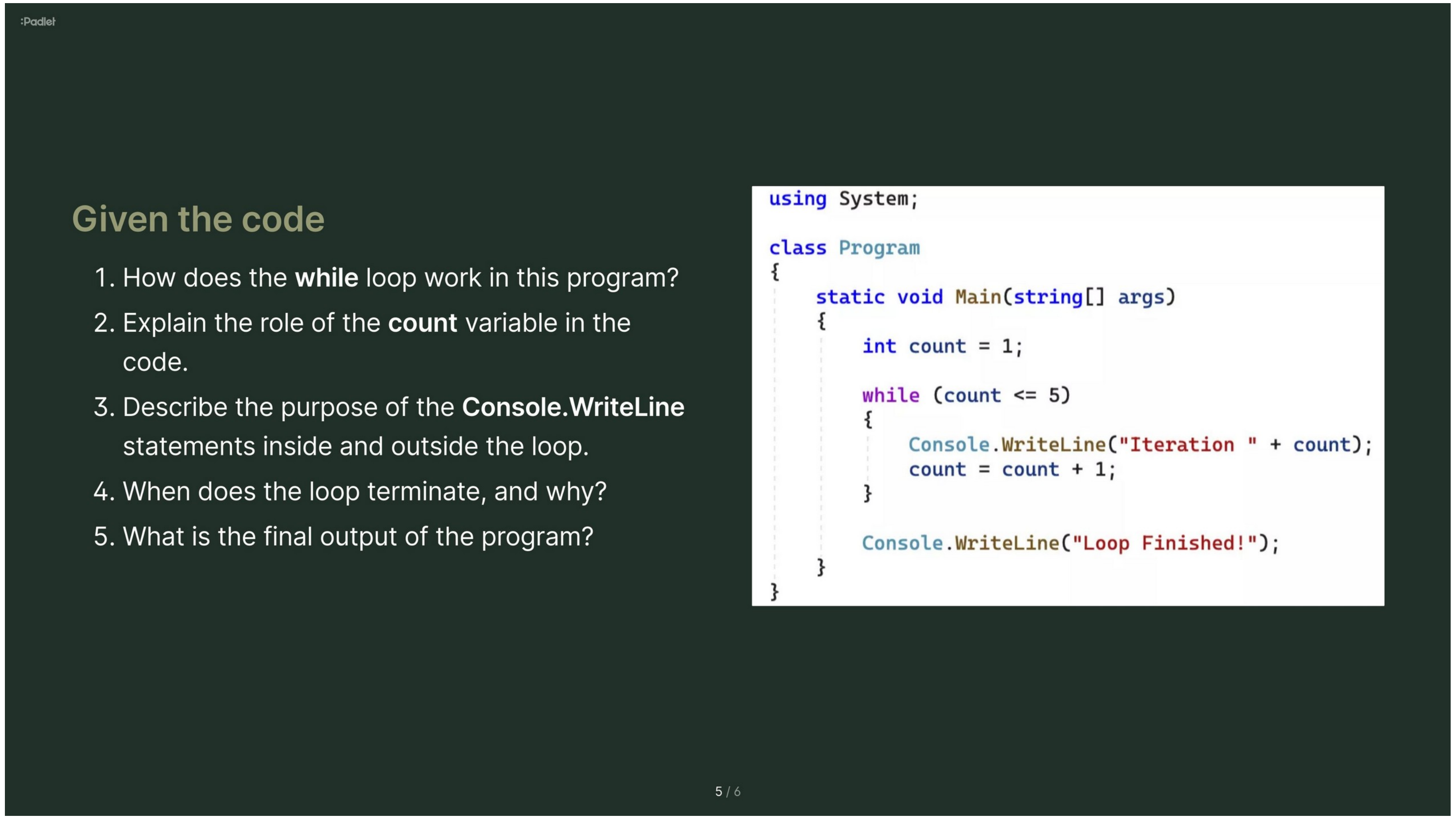
```
static void Main(string[] args)
{
        Console.WriteLine("Number :" + 1);
        Console.WriteLine("Number :" + 2);
        Console.WriteLine("Number :" + 3);
}
```

Printing numbers (up to 1000)

```
static void Main(string[] args)
{
        Console.WriteLine("Number :" + 1);
        Console.WriteLine("Number :" + 2);
        Console.WriteLine("Number :" + 3);
}
```

What if up to **1000**?

Padlet



:Padlet

Given the code

1. How does the **while** loop work in this program?
2. Explain the role of the **count** variable in the code.
3. Describe the purpose of the **Console.WriteLine** statements inside and outside the loop.
4. When does the loop terminate, and why?
5. What is the final output of the program?

```
using System;

class Program
{
    static void Main(string[] args)
    {
        int count = 1;

        while (count <= 5)
        {
            Console.WriteLine("Iteration " + count);
            count = count + 1;
        }

        Console.WriteLine("Loop Finished!");
    }
}
```

5 / 6

What is a loop?

- A loop is a method of structuring statements so that they are **repeated** under certain **conditions**
- The statements that are being executed within the loop are called the **body** of the loop
- Different loops operators are available in C#

While operator

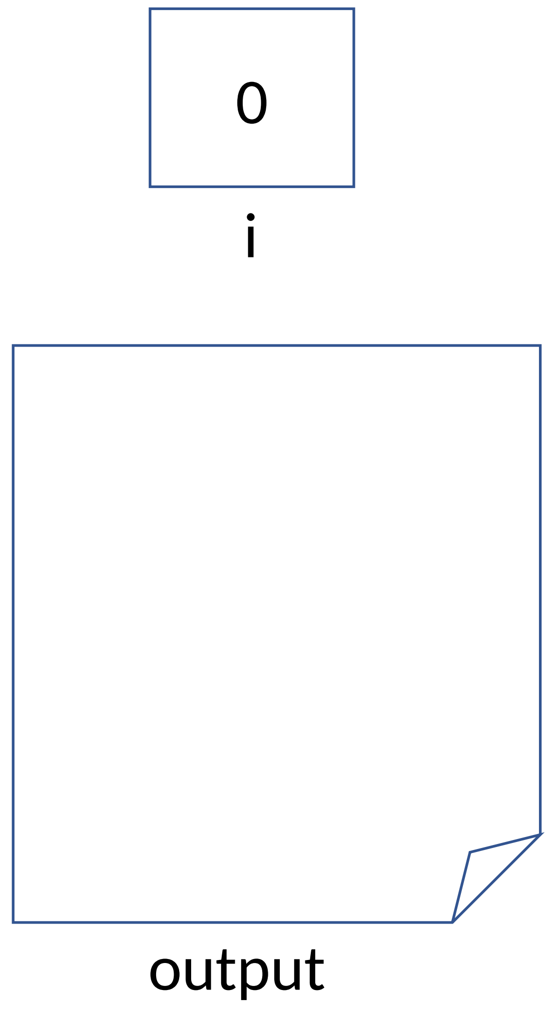
```
while (CONDITION)
{
        STATEMENTS
}
```

While: Printing numbers

```
...
int i = 0; // loop counter
while (i < 3)
{
	Console.WriteLine ("Number: " + (i + 1));
	i = i + 1;
}
Console.WriteLine("Done!");
```

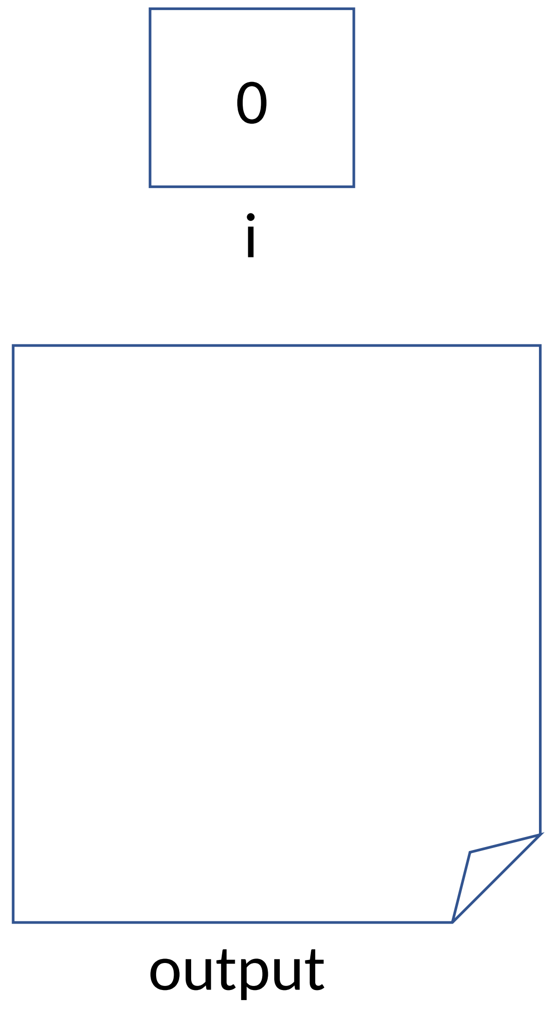
While: Printing numbers

```
...
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```



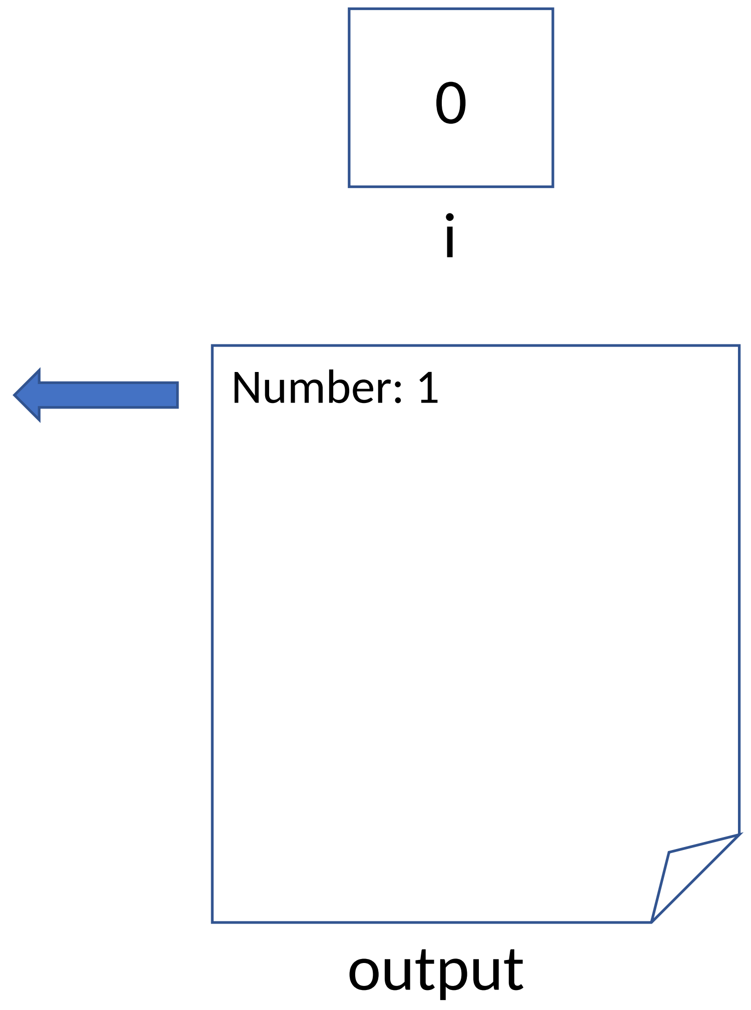
While: Printing numbers

```
...
int i = 0;
while (i < 3)  ←  True
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```



While: Printing numbers

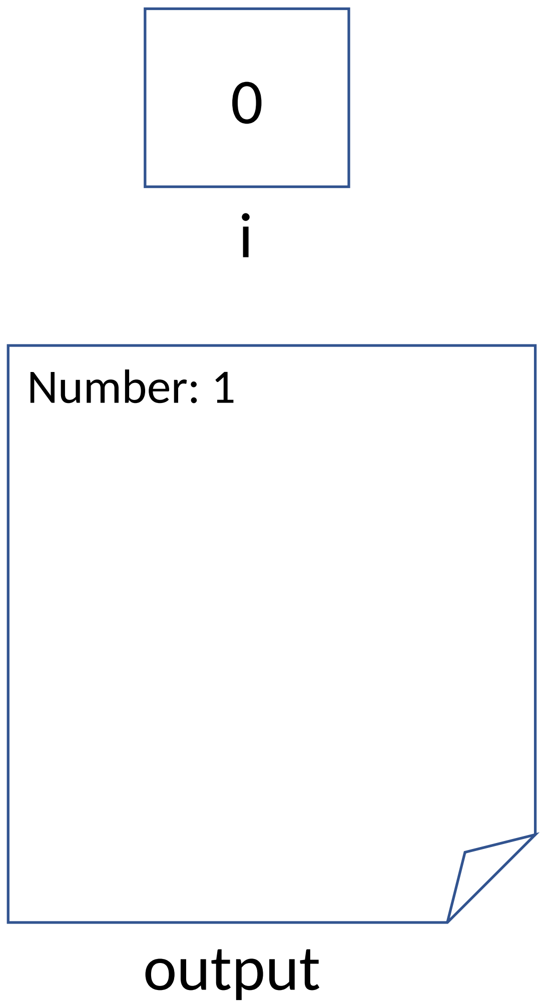
```
...
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```



```
Number: 1
```

While: Printing numbers

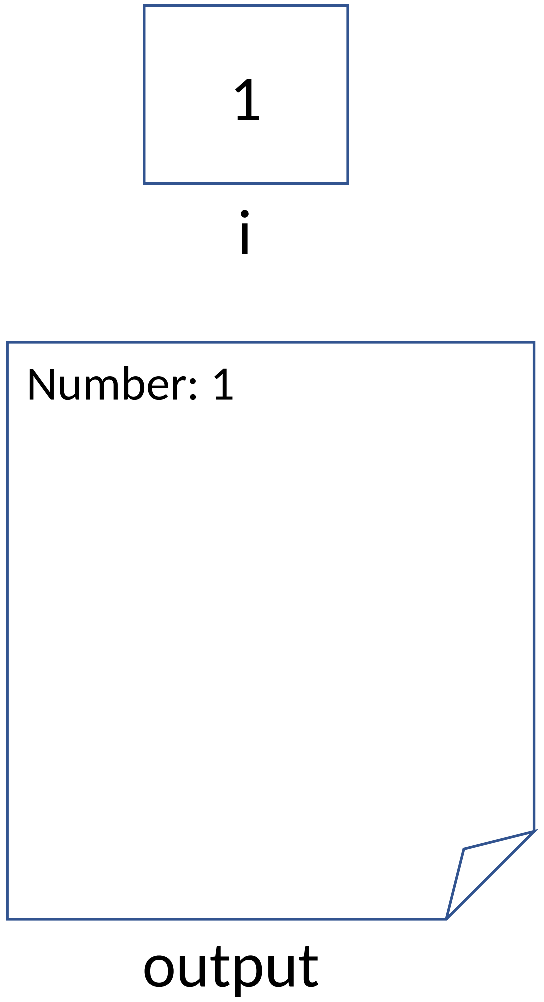
```
...
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```



```
Number: 1
```

While: Printing numbers

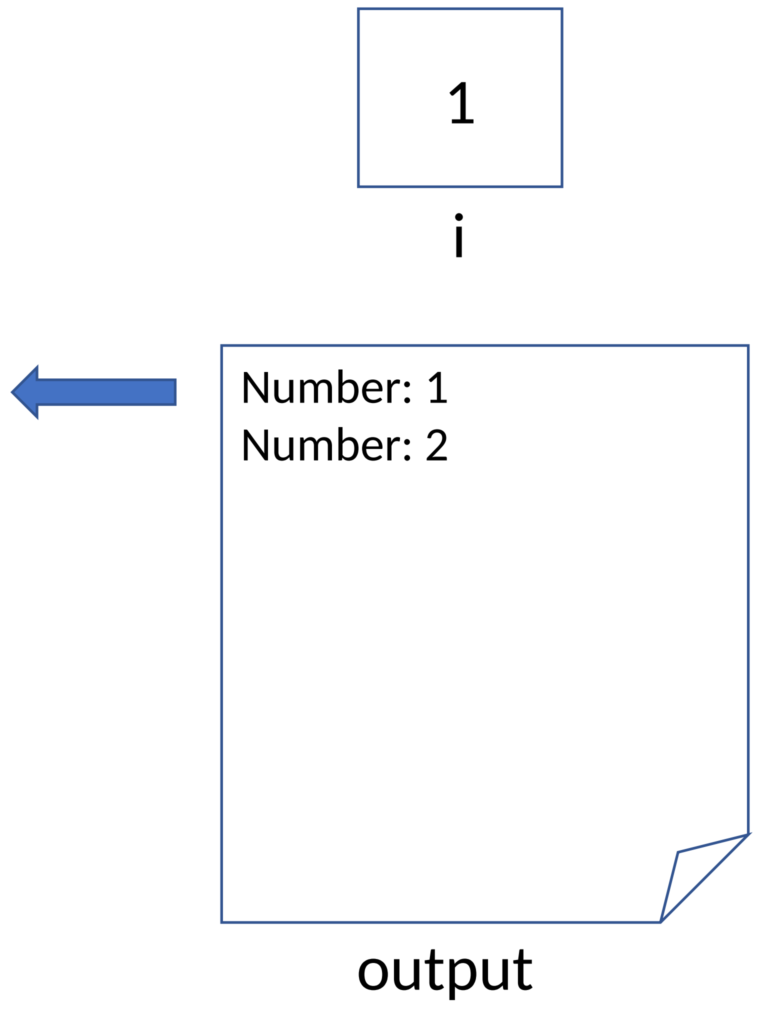
```
...
int i = 0;
while (i < 3)  <━━  True
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```



```
Number: 1
```

While: Printing numbers

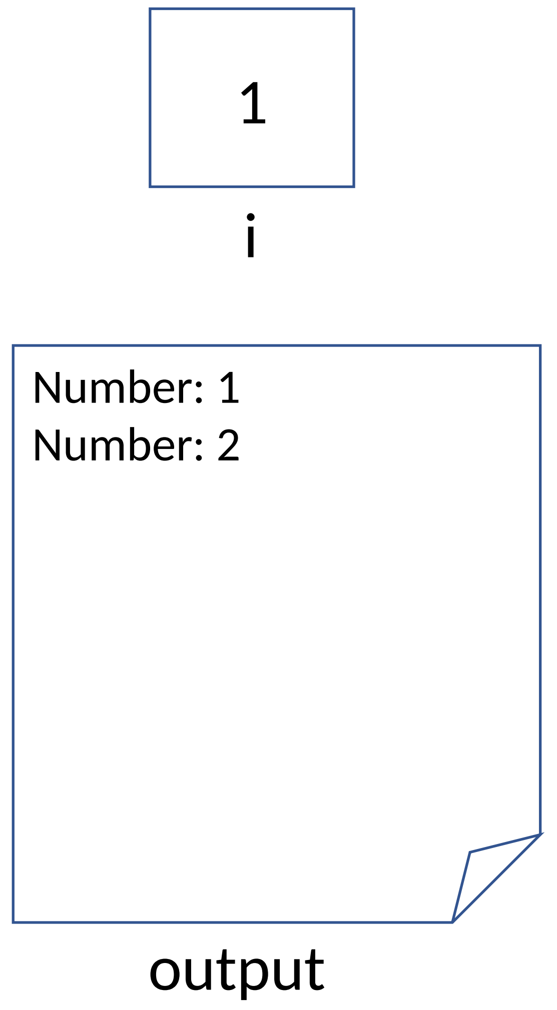
```
...
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```



```
Number: 1
Number: 2
```

While: Printing numbers

```
...
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```

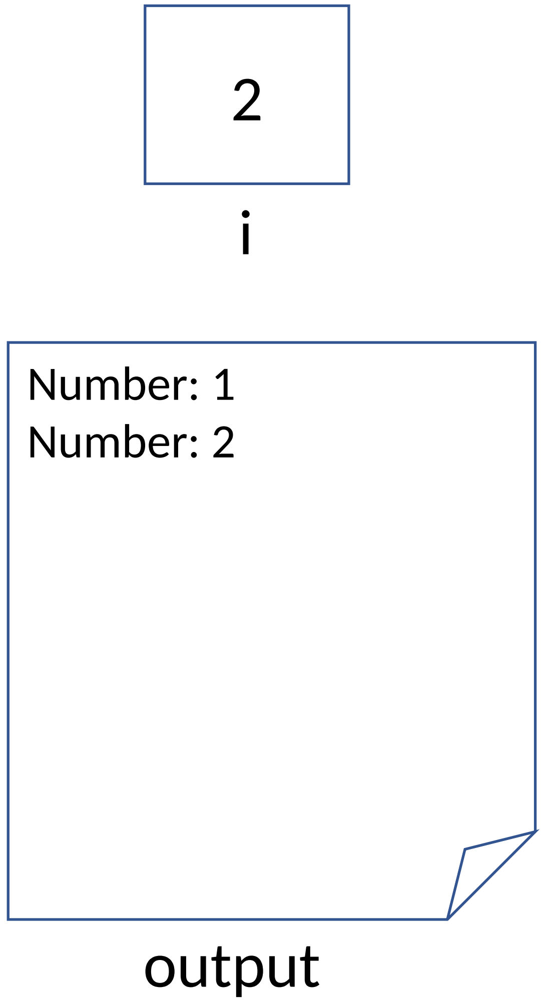


```
Number: 1
Number: 2
```

While: Printing numbers

```
...
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```

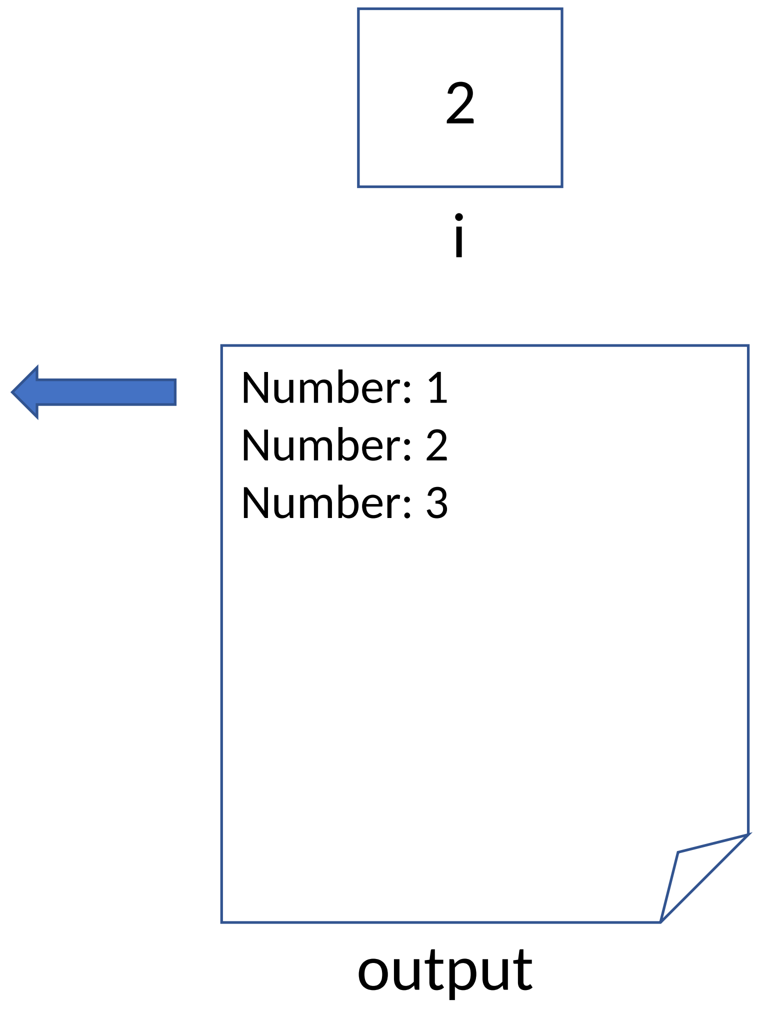
True



```
Number: 1
Number: 2
```

While: Printing numbers

```
...
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```



```
Number: 1
Number: 2
Number: 3
```

While: Printing numbers

```
...
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```

2

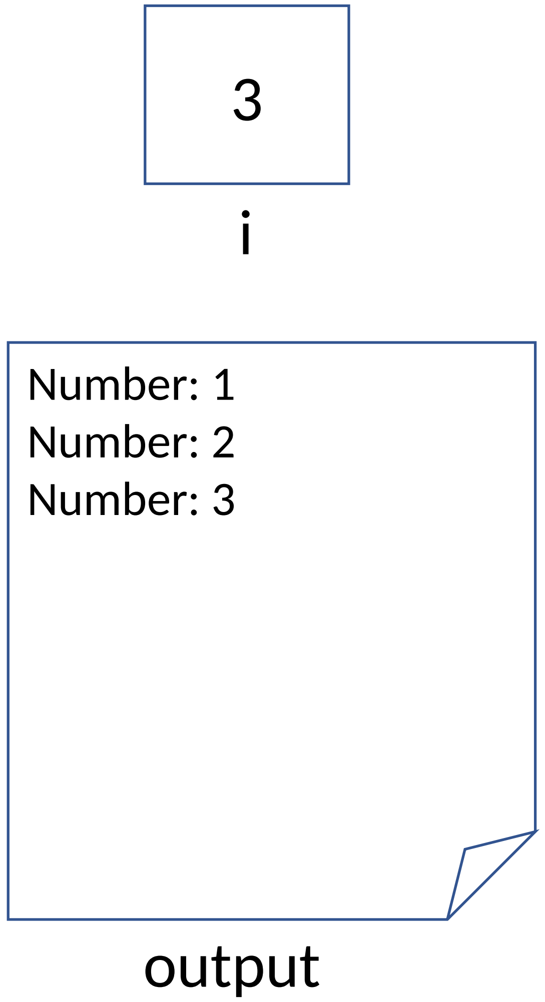
i

```
Number: 1
Number: 2
Number: 3
```

output

While: Printing numbers

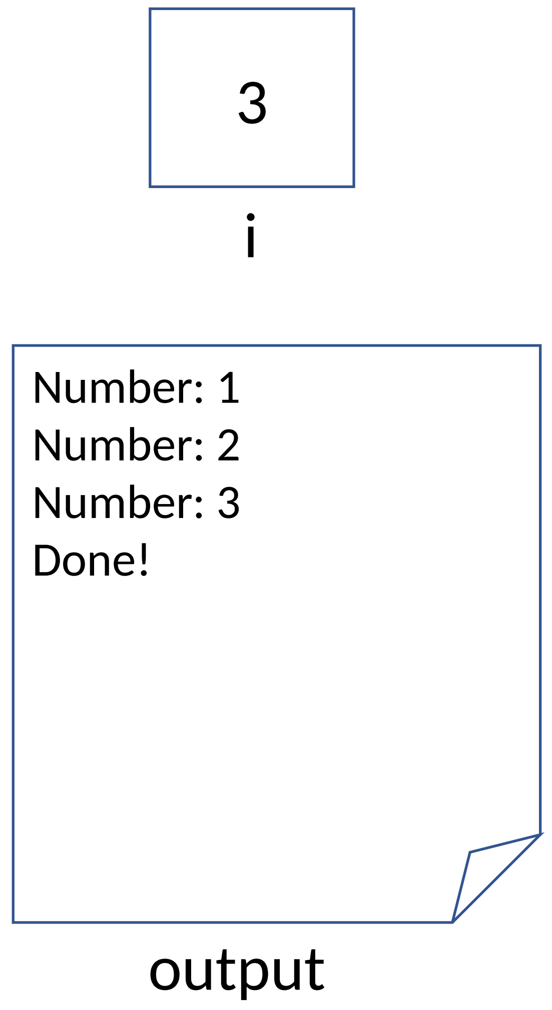
```
…
int i = 0;
while (i < 3)  ⟵  False
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```



```
Number: 1
Number: 2
Number: 3
```

While: Printing numbers

```
...
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```



```
Number: 1
Number: 2
Number: 3
Done!
```

Printing numbers (up to 1000)

```
static void Main(string[] args)
{
    int i = 0;
    while (i < 1000)
    {
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
     }
    Console.WriteLine("Done!");
}
```

What if up to **1000**?

For operator

for (*initialisation*; ***CONDITION***; *update*)

{

    STATEMENTS

}

For operator

```
…
for (int i = 0; i < 3; i = i + 1)
{
		Console.WriteLine ("Number: " + (i + 1));
}
Console.WriteLine("Done!");
```

For vs While operator

```
…
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;
}
Console.WriteLine("Done!");
```

```
…

for (int i = 0; i < 3; i = i + 1)
{
    Console.WriteLine ("Number: " + (i + 1));
}

Console.WriteLine("Done!");
```

Loop remarks

- Always make sure that the loop has a chance to **finish**
- Condition **false** after a **finite** number of iterations

Runs indefinitely because *i* never changes!

```
int i = 0;
while (i < 3)
{
        Console.WriteLine ("Number: " + (i + 1));
}
Console.WriteLine("Done!");
```

Break

```
int i = 0;
while (true)
{
        Console.WriteLine ("Number: " + (i + 1));
        i = i + 1;

        if (i == 3)
        {
            break;
        }
}
Console.WriteLine("Done!");
```

Terminates the closest enclosing iteration statement (or switch statement – later)

Compound Assignment

Given a binary operator *op* (+, -, *, / etc.) and the assignment *'='*

x op= y

is equivalent to

x = x op y

Example

```
int a = 5;
a += 9; // a = a + 9
Console.WriteLine(a); // output: 14
```

Increment and Decrement operators

```
// post-increment
int i = 3;
Console.WriteLine(i);   // output: 3
Console.WriteLine(i++); // output: 3
Console.WriteLine(i);   // output: 4
```

```
// pre-increment
double a = 1.5;
Console.WriteLine(a);   // output: 1.5
Console.WriteLine(++a); // output: 2.5
Console.WriteLine(a);   // output: 2.5
```

```
// post-decrement
double i = 3.0;
Console.WriteLine(i);   // output: 3.0
Console.WriteLine(i--); // output: 3.0
Console.WriteLine(i);   // output: 2.0
```

```
// pre-decrement
int a = 1;
Console.WriteLine(a);   // output: 1
Console.WriteLine(--a); // output: ?
Console.WriteLine(a);   // output: ?
```

Increment and Decrement operators

```
// post-increment
int i = 3;
Console.WriteLine(i);   // output: 3
Console.WriteLine(i++); // output: 3
Console.WriteLine(i);   // output: 4
```

```
// pre-increment
double a = 1.5;
Console.WriteLine(a);   // output: 1.5
Console.WriteLine(++a); // output: 2.5
Console.WriteLine(a);   // output: 2.5
```

```
// post-decrement
double i = 3.0;
Console.WriteLine(i);   // output: 3.0
Console.WriteLine(i--); // output: 3.0
Console.WriteLine(i);   // output: 2.0
```

```
// pre-decrement
int a = 1;
Console.WriteLine(a);   // output: 1
Console.WriteLine(--a); // output: 0
Console.WriteLine(a);   // output: 0
```

What does this code do?

```
int sum = 0;
for (int number = 1; number < 21; number++)
{
  if (number % 3 == 0)
  {
    sum += number;
  }
}
Console.WriteLine($"The sum is {sum}");
```

replaces
number = number + 1

Poll on While Loops

PollEveryWhere: https://pollev.com/francescotusa

What does this code do?

```
int sum = 0;
for (int number = 1; number < 21; number++)
{
  if (number % 3 == 0)
  {
    sum += number;
  }
}
Console.WriteLine($"The sum is {sum}");
```

replaces
number = number + 1

sum = 3 + 6 + 9 + 12 + 15 + 18 = 63

Outline

- Summary of the previous lecture
- Loops
- More on selection statements: switch-case

Selection statements: **if-else-if**

```
if (CONDITION1) {
        STATEMENTS1
} else if (CONDITION2) {
        STATEMENTS2
} else if (CONDITION3) {
        STATEMENTS3
} else {
        STATEMENTS
}
```

Selection statements: **if-else-if**

```
if (CONDITION1) {
        STATEMENTS1
} else if (CONDITION2) {
        STATEMENTS2
} else if (CONDITION3) {
        STATEMENTS3
} else {
        STATEMENTS
}
```

What if we have *10 CONDITIONS*?

- Readability
- Performance

Selection statements: **switch**

```
switch (EXPRESSION) {
        case PATTERN:
        STATEMENTS
        case PATTERN :
        STATEMENTS
        default:
        STATEMENTS
}
```

A ***STATEMENTS*** list is executed based on a *pattern match* with a *match expression*

Switch statement: example

```
static void Main(string[] args)
{
    int temperature = Convert.ToInt32(Console.ReadLine());

    switch (temperature)
    {
        case 30:
            Console.WriteLine("Critical");
            break;

        case 24:
            Console.WriteLine("Warning");
            break;

        default:
            Console.WriteLine("Normal");
            break;
    }
}
```

Historically (pre C# 9), only *constant patterns* were allowed (e.g., case 30:)

Switch statement: example

```
static void Main(string[] args)
{
   int temperature = Convert.ToInt32(Console.ReadLine());

   switch (temperature)
   {
      case > 30:
         Console.WriteLine("Critical");
         break;

      case > 24:
         Console.WriteLine("Warning");
         break;

      default:
         Console.WriteLine("Normal");
         break;
   }
}
```

Relational pattern matching

Switch statement: example

```
static void Main(string[] args)
{
   int temperature = Convert.ToInt32(Console.ReadLine());

   switch (temperature)
   {
      case > 30:
         Console.WriteLine("Critical");
         break;

      case > 24:
         Console.WriteLine("Warning");
         break;

      default:
         Console.WriteLine("Normal");
         break;
   }
}
```

1

2

Matching is performed in text order

Switch statement: example

```
static void Main(string[] args)
{
    int temperature = Convert.ToInt32(Console.ReadLine());

    switch (temperature)
    {
        case > 30:
            Console.WriteLine("Critical");
            break;

        case > 24:
            Console.WriteLine("Warning");
            break;

        default:
            Console.WriteLine("Normal");
            break;
    }
}
```

statements to execute when a match expression does **NOT** match any other case pattern

Switch statement: example

```
static void Main(string[] args)
{
   int temperature = Convert.ToInt32(Console.ReadLine());

   switch (temperature)
   {
      case > 30:
         Console.WriteLine("Critical");
         break;

      case > 24:
         Console.WriteLine("Warning");
         break;

      default:
         Console.WriteLine("Normal");
         break;
   }
}
```

Within a switch statement, control cannot fall through from one switch section to the next

Switch statement: non-valid data

- So far we have:
  - Critical: **T > 30 C**
  - Warning: **24 C < T <= 30 C**
  - Normal: **T <= 24 C**
- Check also if data is not valid (e.g., out of an expected range):
  - **T >= 100 C**

Switch statement: unreachable cases

- The compiler generates an **error** when a *switch* statement contains an unreachable *case*
- That is a *case* that is already handled by an upper *case* or whose pattern is **impossible** to match

Switch statement example: invalid data

```
switch (temperature)
{
      case >= 100:
            Console.WriteLine("Not Valid!");
            break;

      case > 30:
            Console.WriteLine("Critical");
            break;

      case > 24:
            Console.WriteLine("Warning");
            break;

      default:
            Console.WriteLine("Normal");
            break;
          }
}
```

A

```
switch (temperature)
{
      case > 30:
            Console.WriteLine("Critical");
            break;

      case > 24:
            Console.WriteLine("Warning");
            break;

      case >= 100:
            Console.WriteLine("Not Valid!");
            break;

      default:
            Console.WriteLine("Normal");
            break;
          }
}
```

B

Poll on Switch Case

PollEveryWhere: https://pollev.com/francescotusa

Switch statement example: invalid data

case >=100 is an unreachable case, values already handled by an upper case

For instance, temperature = 110 will always be > 30

```
switch (temperature)
{
    case > 30:
        Console.WriteLine("Critical");
        break;

    case > 24:
        Console.WriteLine("Warning");
        break;

    case >= 100:
        Console.WriteLine(”Not Valid!");
        break;

    default:
        Console.WriteLine("Normal");
        break;
      }
}
```

B

Switch statement: example

```
static void Main(string[] args)
{
   int temperature = Convert.ToInt32(Console.ReadLine());

   switch (temperature)
   {
       case >= 100:
       case < 0:
         Console.WriteLine("Not Valid!");
         break;
       ...
   }
}
```

Different *case* labels / patterns at the beginning of a section

Statements in that section will run if *any* of those cases apply

What logical operation is this similar to?

Switch statement: example

```
static void Main(string[] args)
{
   int temperature = Convert.ToInt32(Console.ReadLine());

   switch (temperature)
   {
       case >= 100 when temperature <= 110:
       case < 0:
         Console.WriteLine("Not Valid!");
         break;
       ...
   }
}
```

Additional conditions can be introduced via case guards using *when*

What logical operation is this similar to?