7

Control Statements

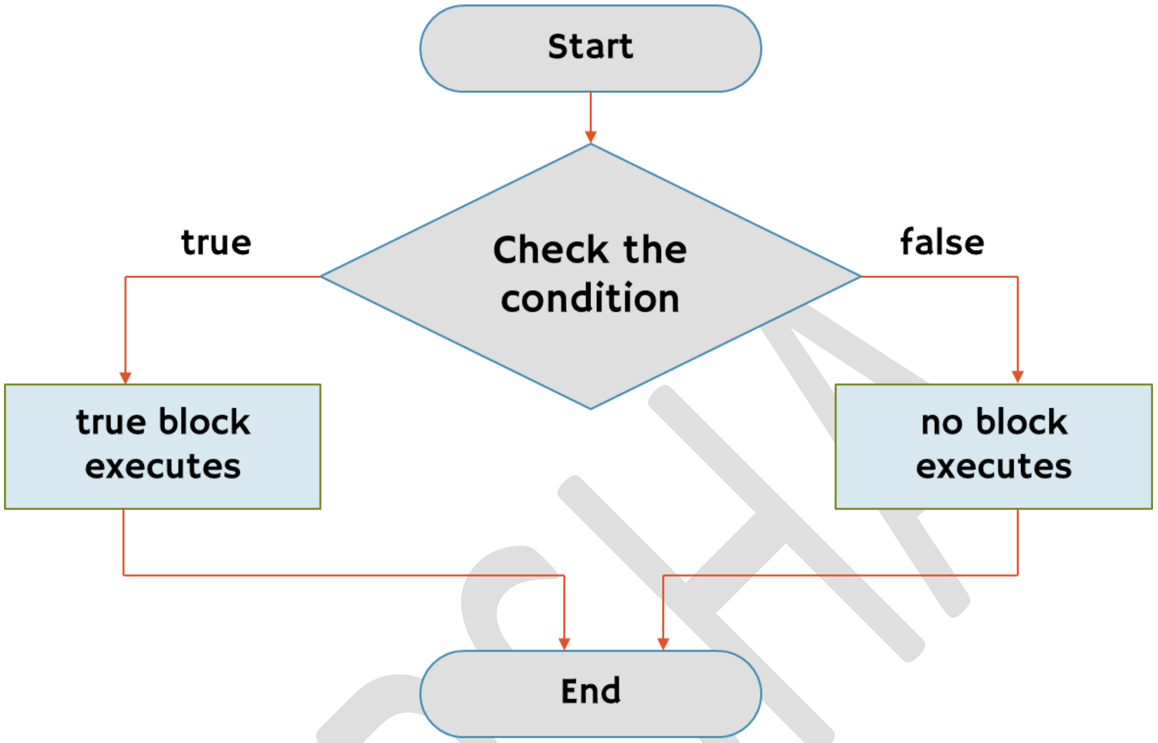
What are Control Statements?

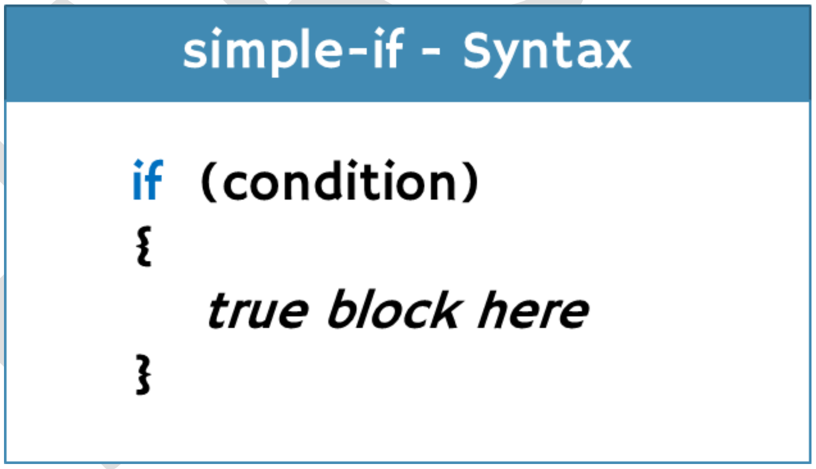
- Control Statements are used to control the program execution flow.
- Used to make the execution flow "jump forward" or "jump backward".

Classification of Control Statements

- **Conditional Control Statements**
  - if (simple-if, if-else, else-if, nested-if)
  - switch-Case
- **Looping Control Statements**
  - while
  - do-While
  - for
- **Jumping Control Statements**
  - goto
  - break
  - continue

Simple-if



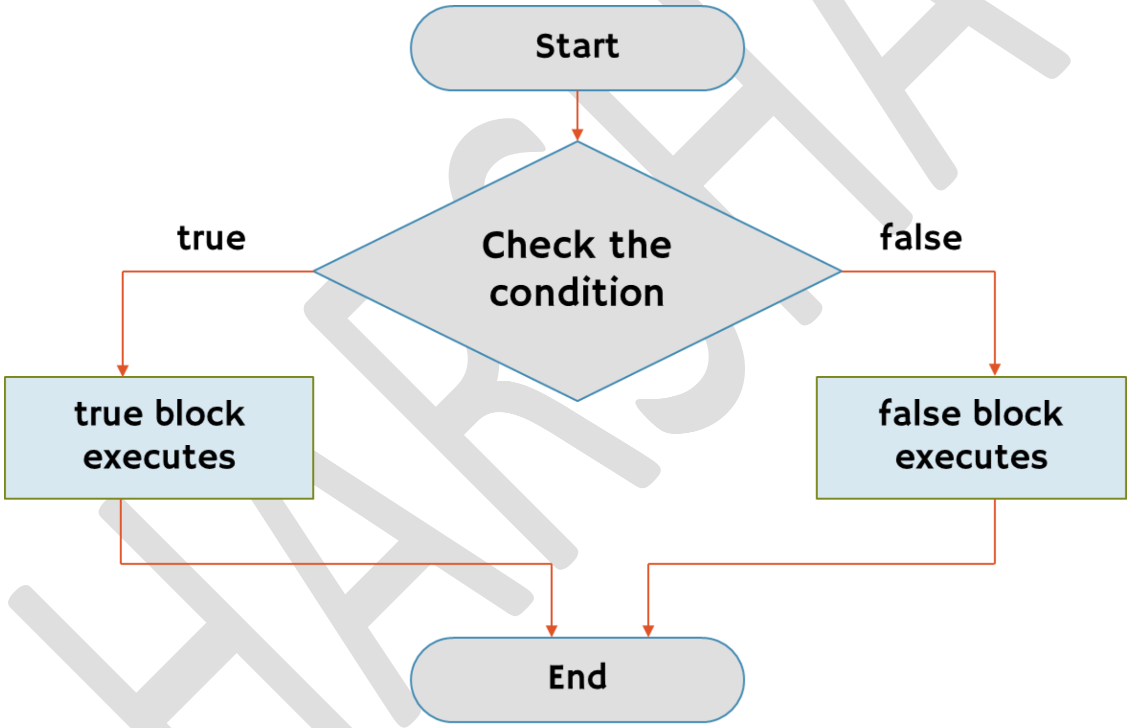

simple-if - Syntax

```
if (condition)
{
    true block here
}
```

simple-if - Example

```
if (x < 10)
{
    System.Console.WriteLine("x is larger");
}
```

If-else



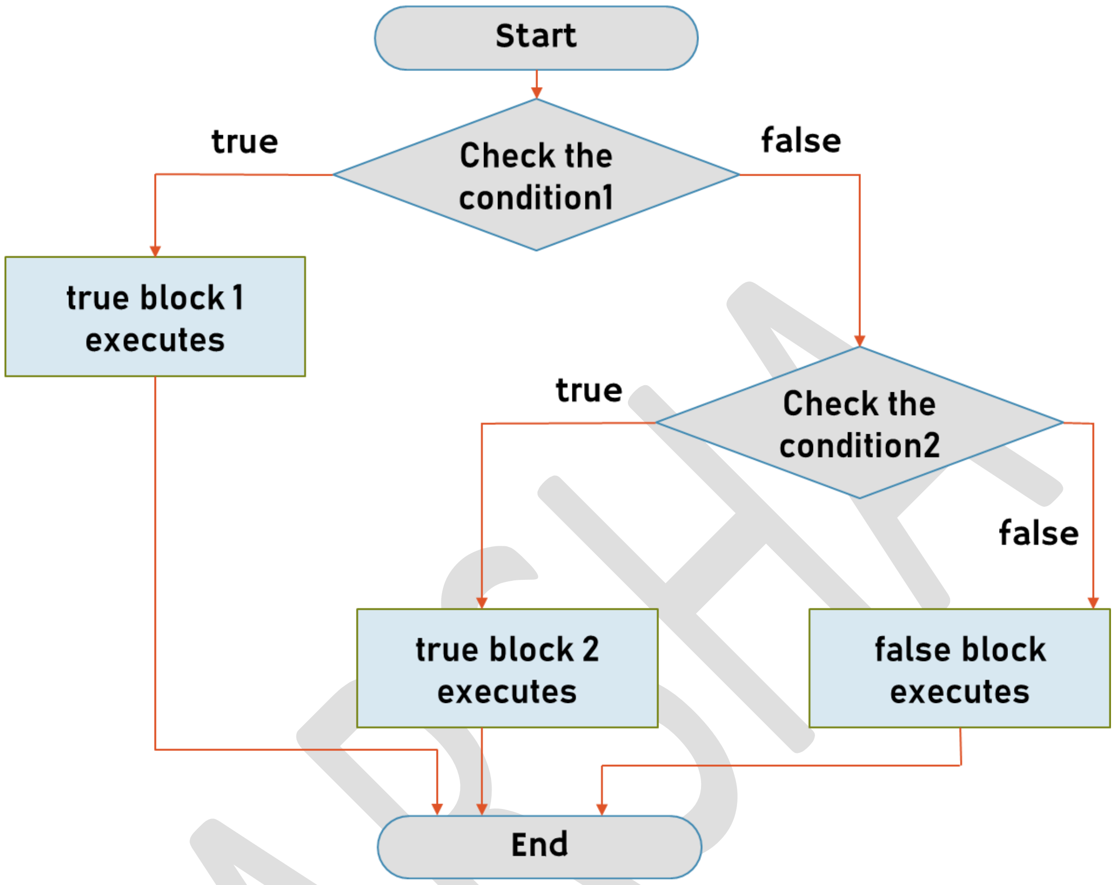
if - else - Syntax

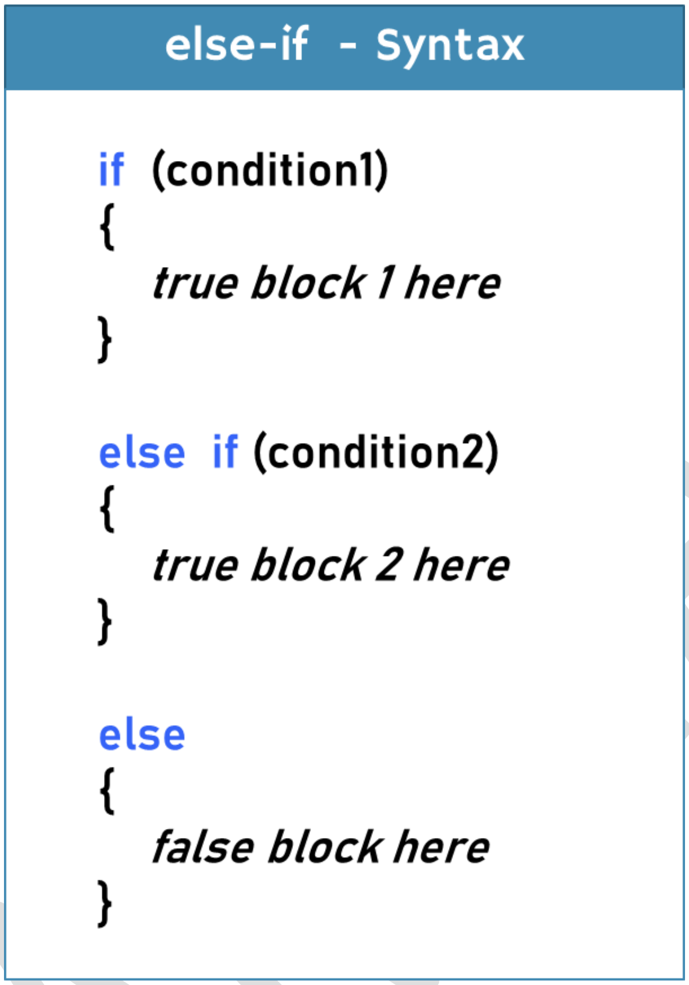
```
if (condition)
{
    true block here
}
else
{
    false block here
}
```

if-else - Example

```
if (x > 10)
{
    System.Console.WriteLine("x is larger");
}
else
{
    System.Console.WriteLine("x is smaller");
}
```

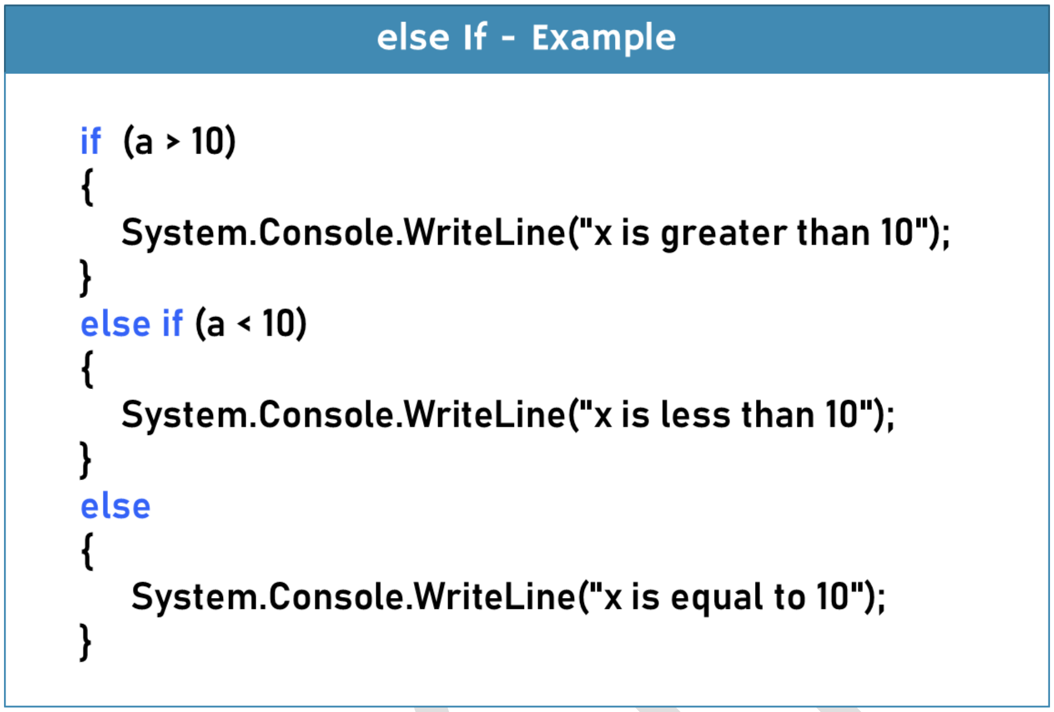
Else-if


Start
true
Check the condition1
false
true block 1 executes
true
Check the condition2
false
true block 2 executes
false block executes
End



else-if - Syntax

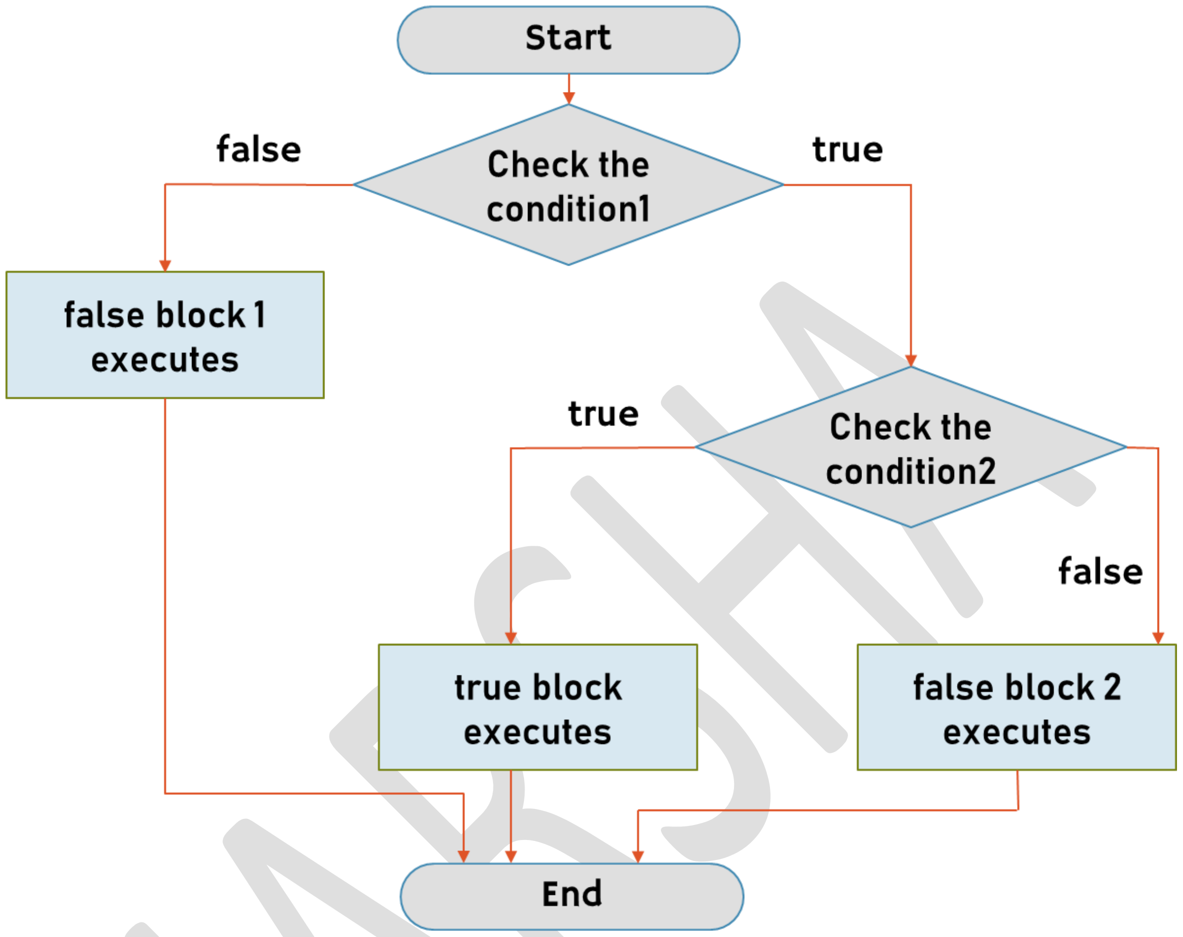
```
if  (condition1)
{
    true block 1 here
}

else  if (condition2)
{
    true block 2 here
}

else
{
    false block here
}
```

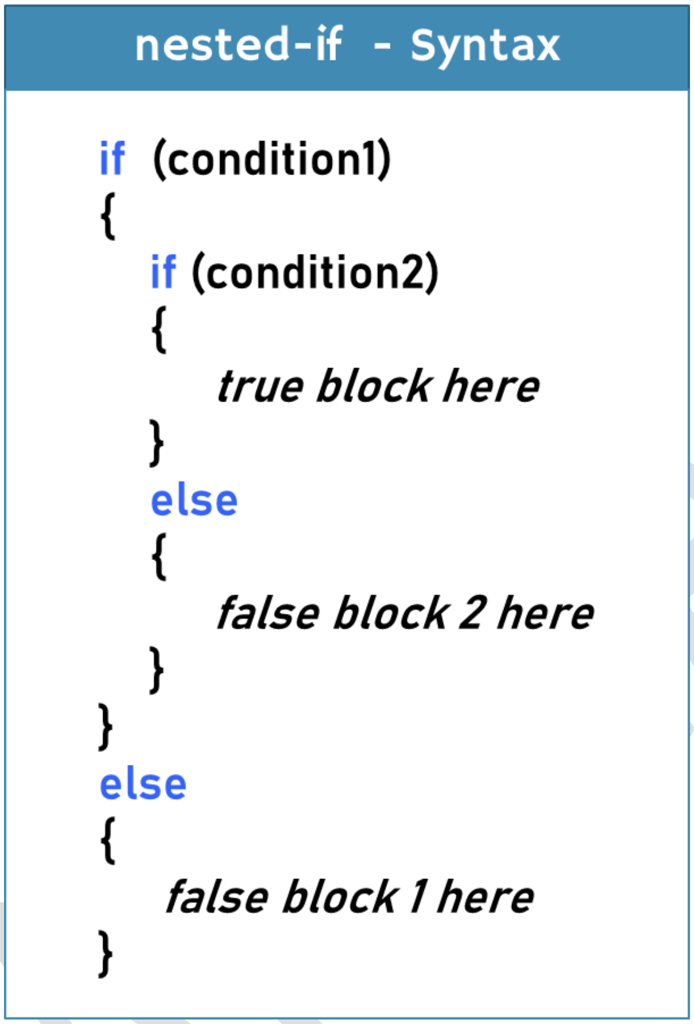


else If - Example

```
if  (a > 10)
{
   System.Console.WriteLine("x is greater than 10");
}
else if (a < 10)
{
   System.Console.WriteLine("x is less than 10");
}
else
{
    System.Console.WriteLine("x is equal to 10");
}
```

Nested if


Start
false
Check the condition1
true
false block 1 executes
true
Check the condition2
false
true block executes
false block 2 executes
End



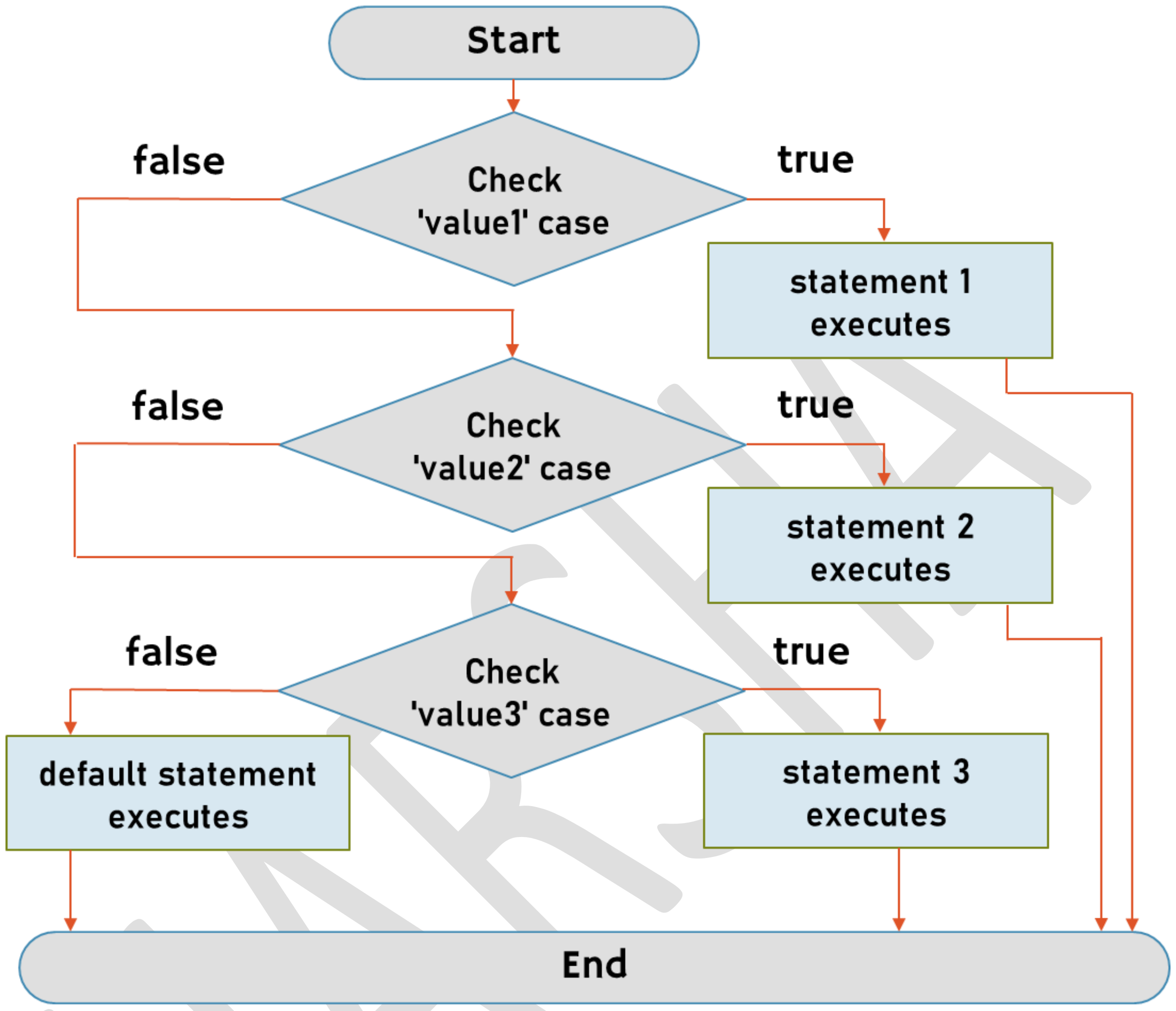
nested-if - Syntax

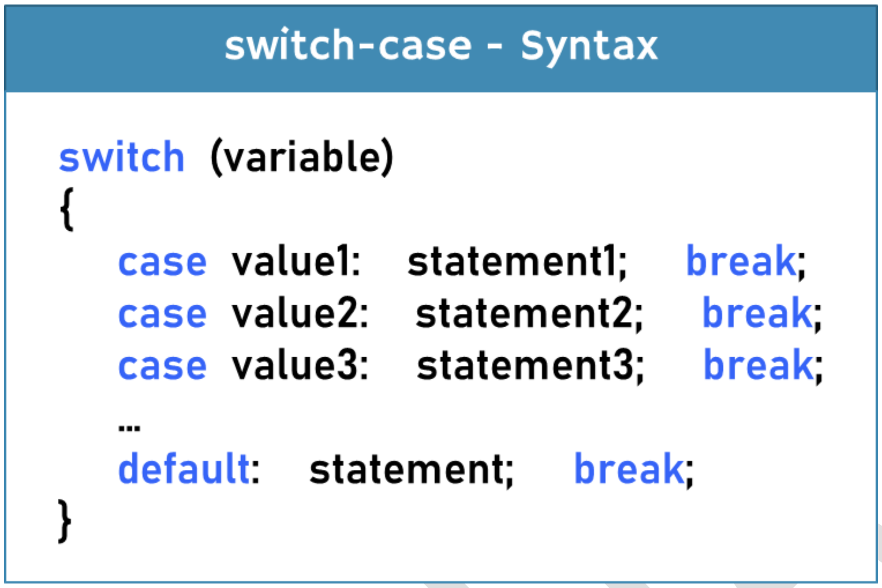
```
if (condition1)
{
    if (condition2)
    {
        true block here
    }
    else
    {
        false block 2 here
    }
}
else
{
    false block 1 here
}
```

nested If - Example

```
if  (a >= 10)
{
    if (a > 10)
    {
        System.Console.WriteLine("x is greater than 10");
    }
    else
    {
        System.Console.WriteLine("x is equal to 10");
    }
}
else
{
    System.Console.WriteLine("x is less than 10");
}
```

Switch-case

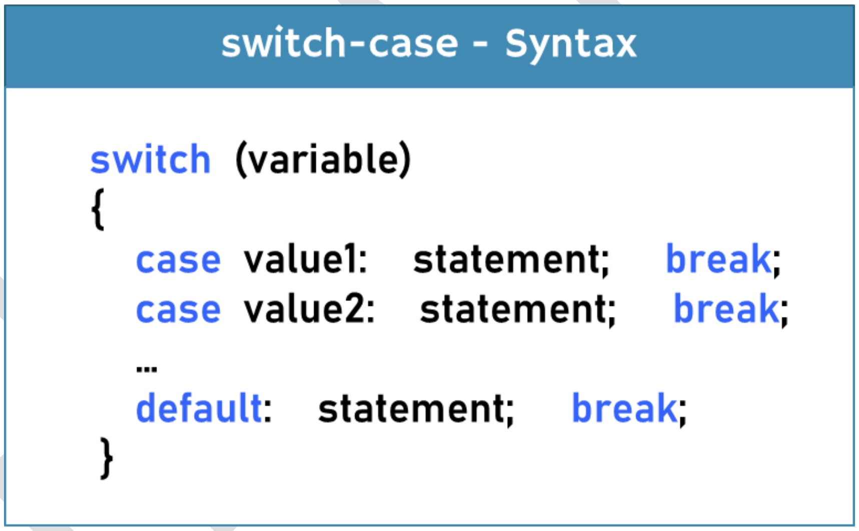

Start
Check 'value1' case
false
true
statement 1 executes
Check 'value2' case
false
true
statement 2 executes
Check 'value3' case
false
true
default statement executes
statement 3 executes
End



switch-case - Syntax

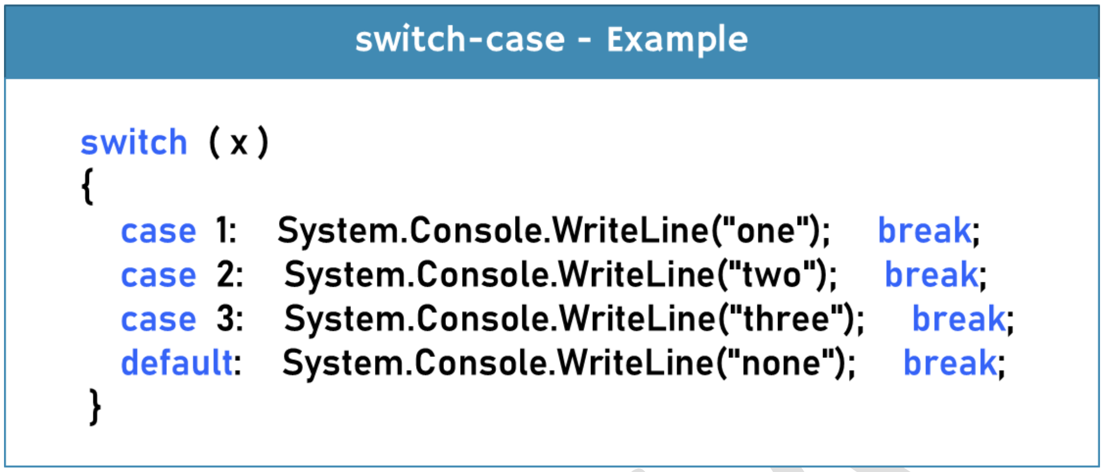
```
switch  (variable)
{
    case  value1:   statement1;    break;
    case  value2:   statement2;    break;
    case  value3:   statement3;    break;
    ...
    default:   statement;    break;
}
```

Switch-case



switch-case - Syntax

```
switch  (variable)
{
   case  value1:   statement;    break;
   case  value2:   statement;    break;
   ...
   default:   statement;    break;
}
```

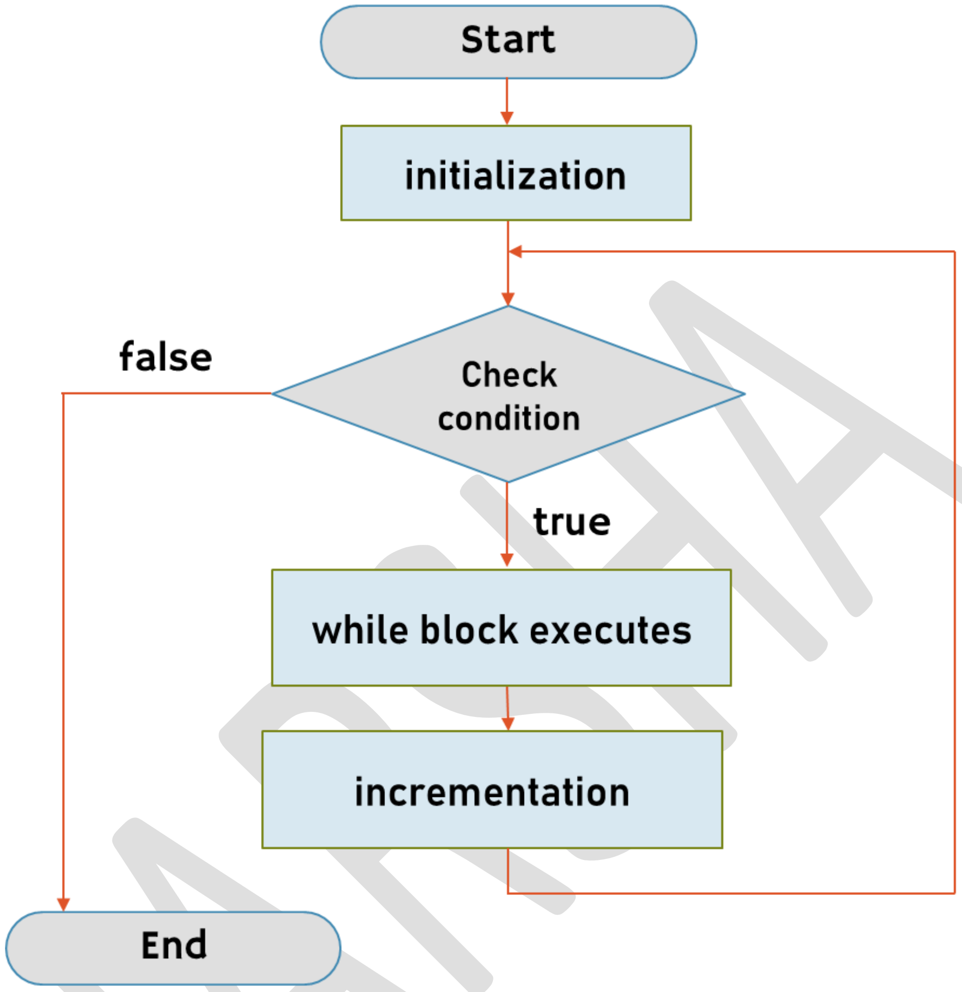


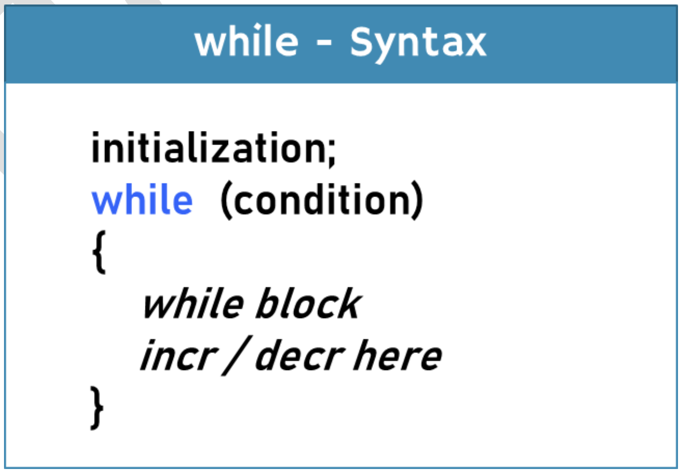
switch-case - Example

```
switch ( x )
{
   case 1:   System.Console.WriteLine("one");    break;
   case 2:   System.Console.WriteLine("two");    break;
   case 3:   System.Console.WriteLine("three");    break;
   default:   System.Console.WriteLine("none");    break;
}
```

- Used to check a variable value, many times, whether it matches with any one of the list of values.
- Among all cases, only one will execute.
- If all cases are not matched, it executes the "default case".

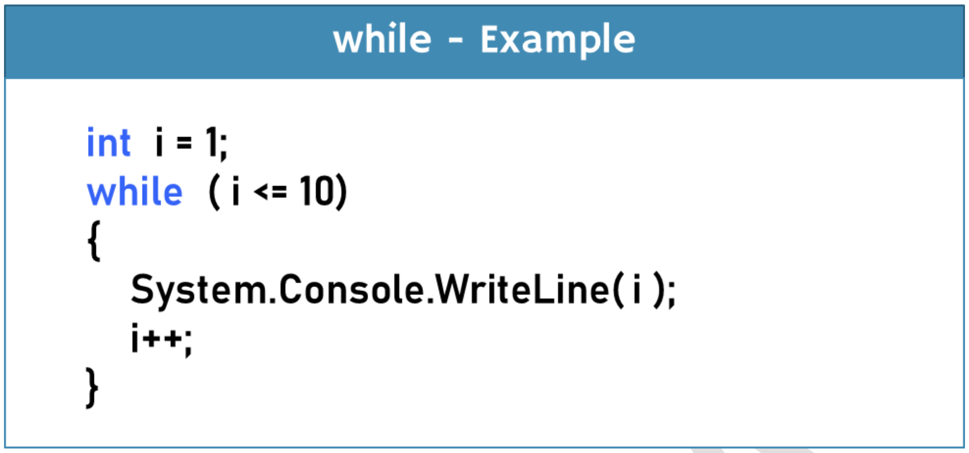
While





while - Syntax

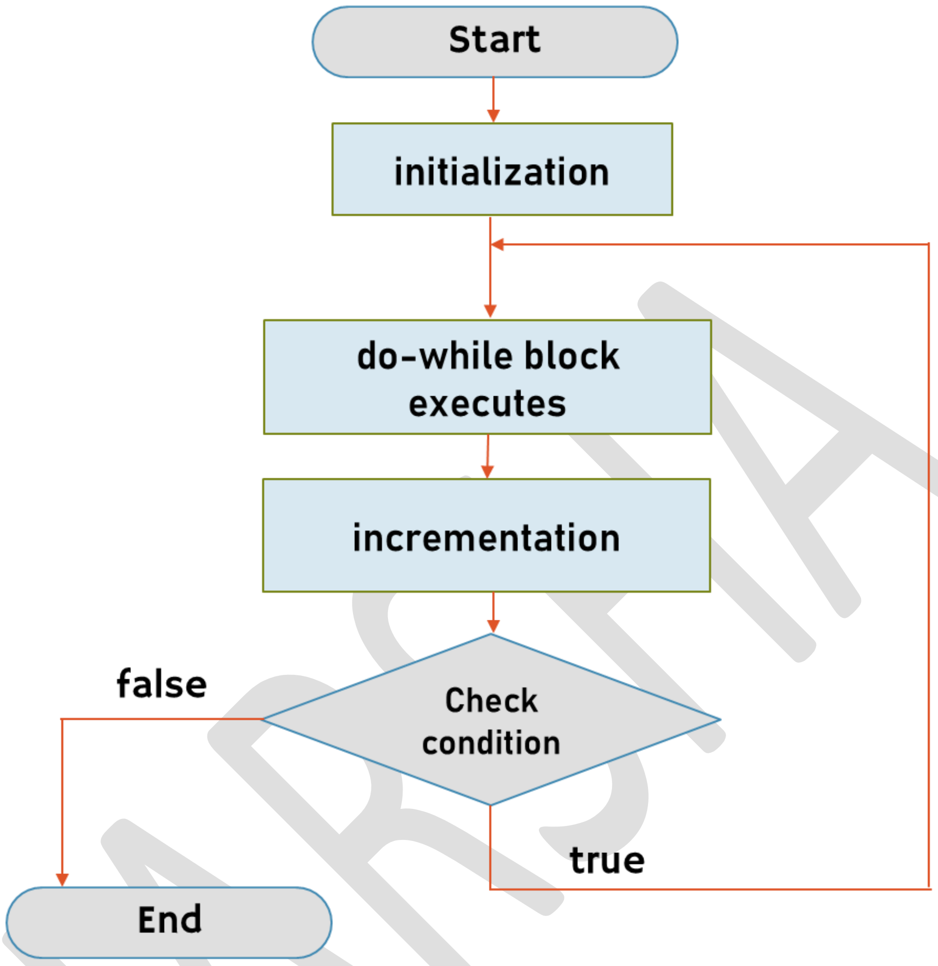
```
initialization;
while (condition)
{
    while block
    incr / decr here
}
```

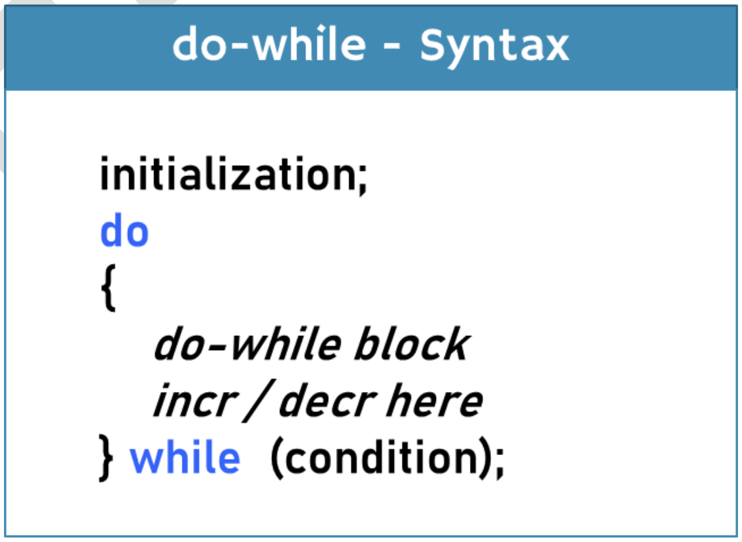

while - Example

```
int  i = 1;
while  ( i <= 10)
{
   System.Console.WriteLine( i );
   i++;
}
```

- Used to execute a set of statements, as long as the condition is TRUE.
- Once the condition is false, it will exit from the while loop.

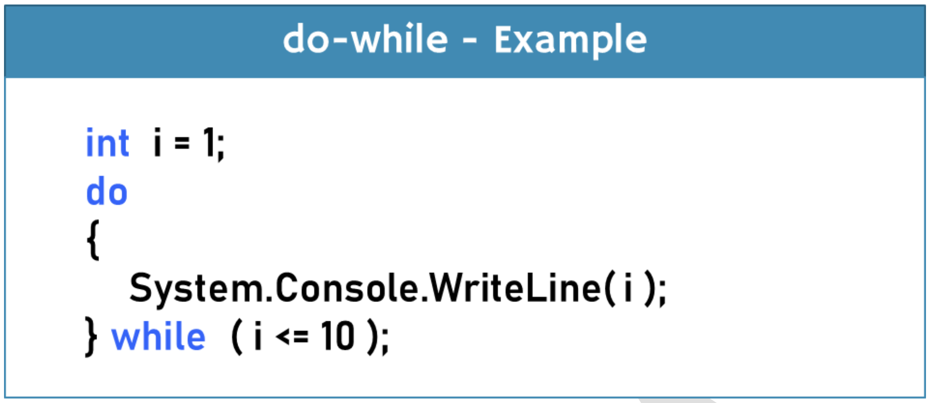
Do-While





do-while - Syntax

```
initialization;
do
{
    do-while block
    incr / decr here
} while  (condition);
```

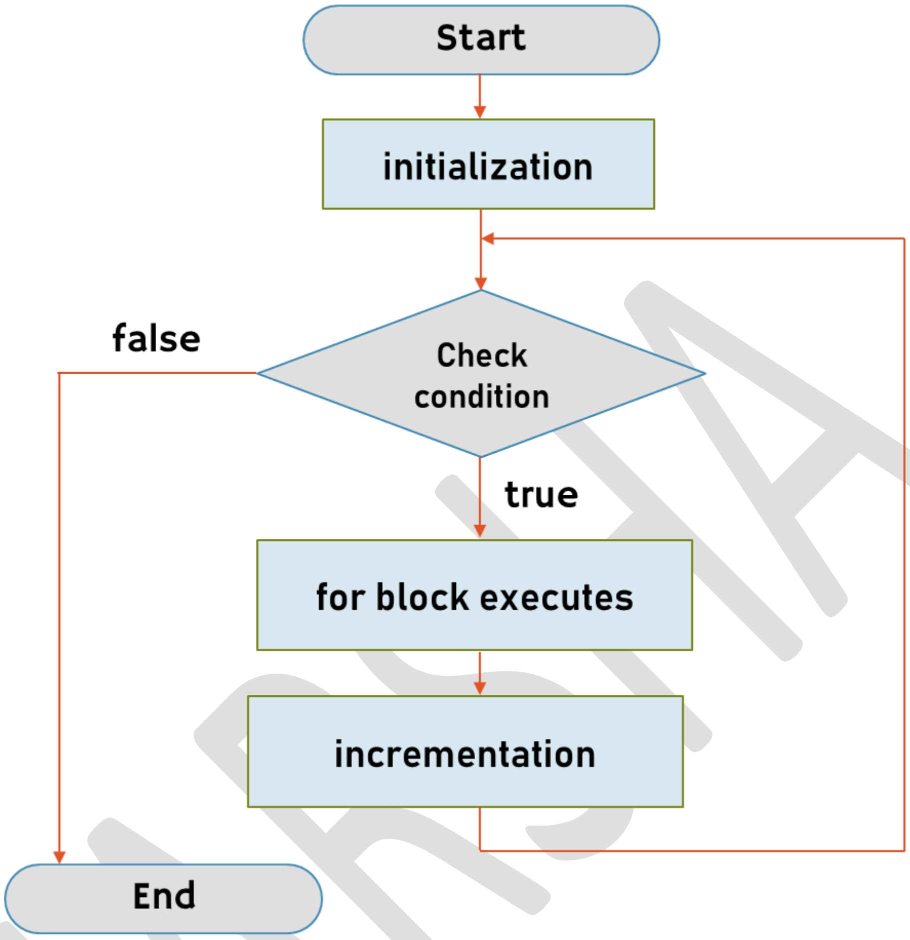


do-while - Example

```
int  i = 1;
do
{
   System.Console.WriteLine( i );
} while  ( i <= 10 );
```

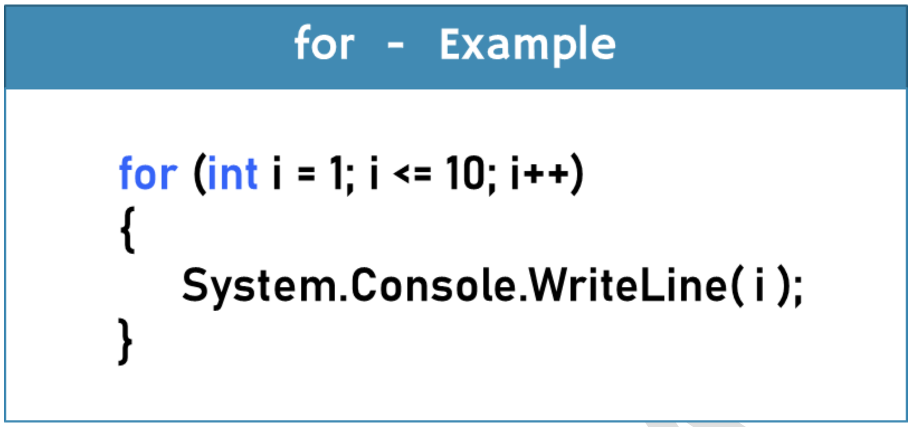
- Used to execute a set of statements, as long as the condition is TRUE.
- Once the condition is false, it will exit from the while loop.
- It is same as "While loop"; but the difference is:
  - It executes at least one time even though the condition is false, because it doesn't check the condition for the first time.
  - Second time onwards, it is same as "while" loop.

For



for - Syntax

```
for (initialization; condition; incrementation)
{
        for block
}
```

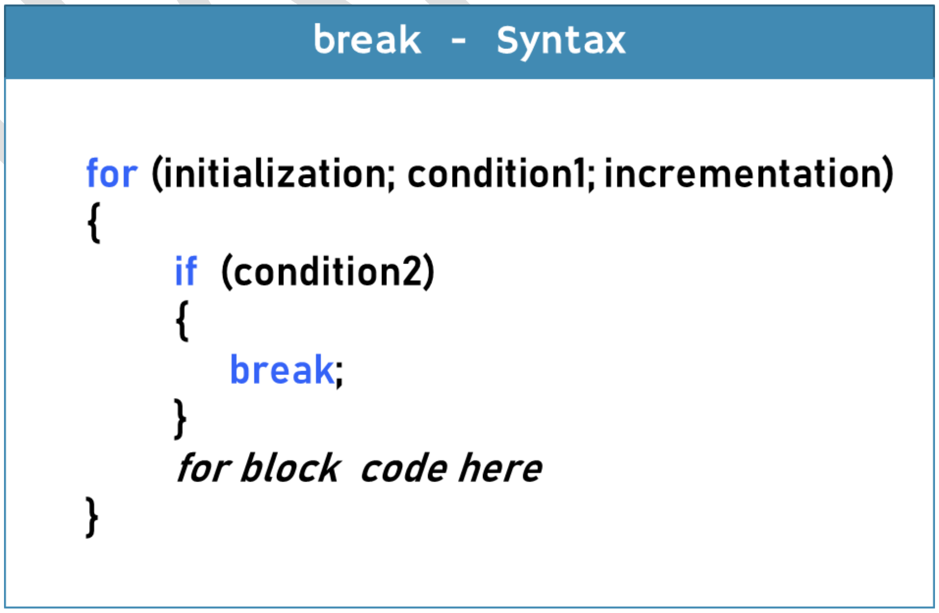


for - Example

```
for (int i = 1; i <= 10; i++)
{
     System.Console.WriteLine( i );
}
```

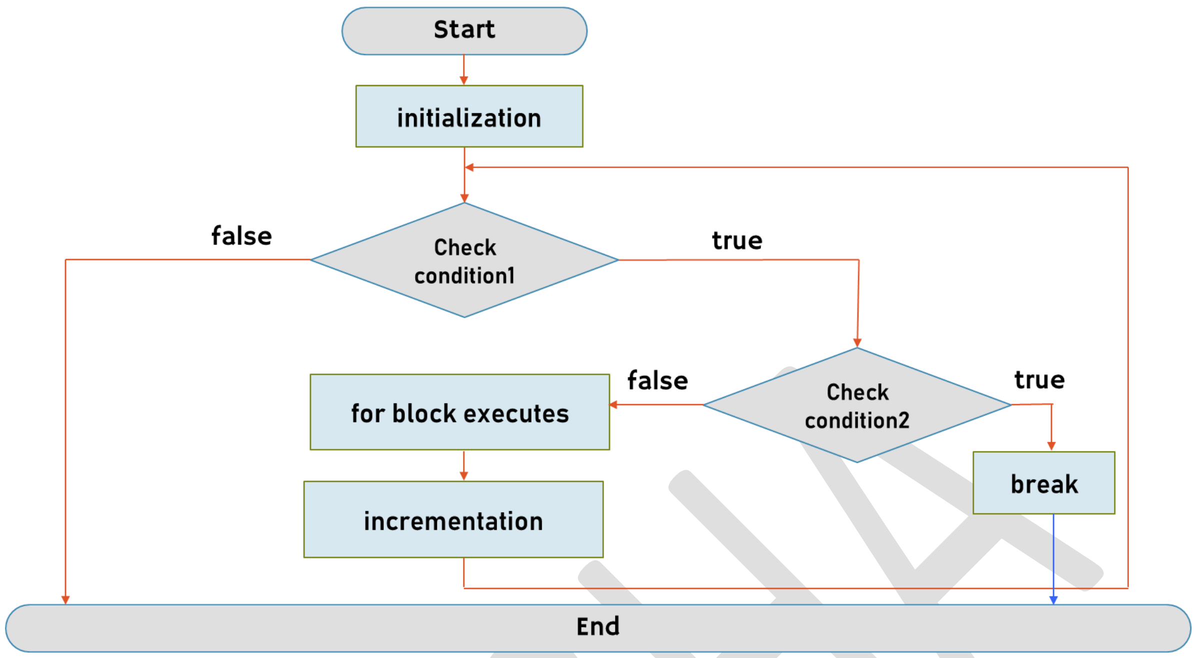
- Used to execute a set of statements, as long as the condition is TRUE.
- Once the condition is false, it will exit from the while loop.
- It is same as "While loop"; but the difference is:
  - We can write all loop details (initialization, condition, incrementation), in-one-line.

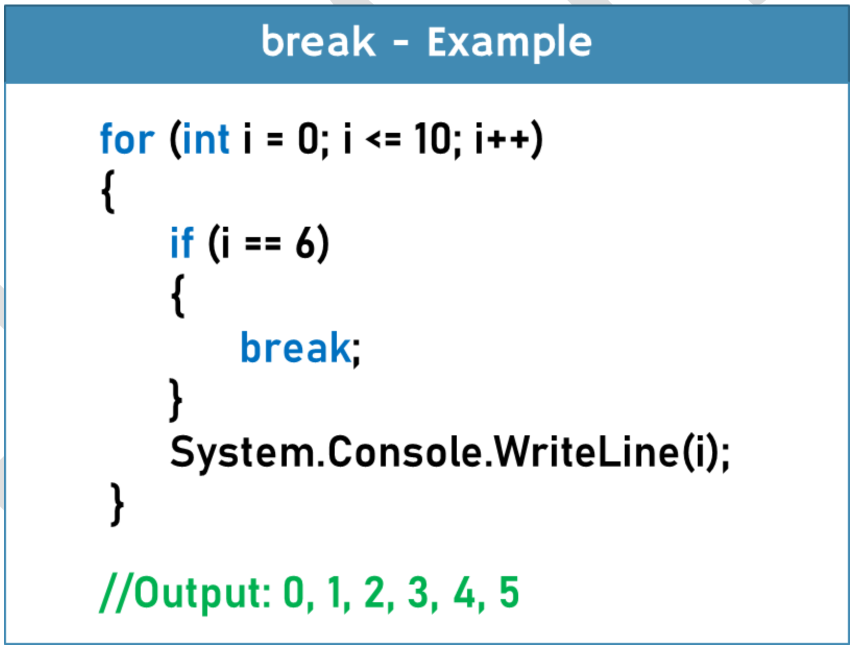
Break



break - Syntax

```
for (initialization; condition1; incrementation)
{
     if  (condition2)
     {
         break;
     }
     for block  code here
}
```



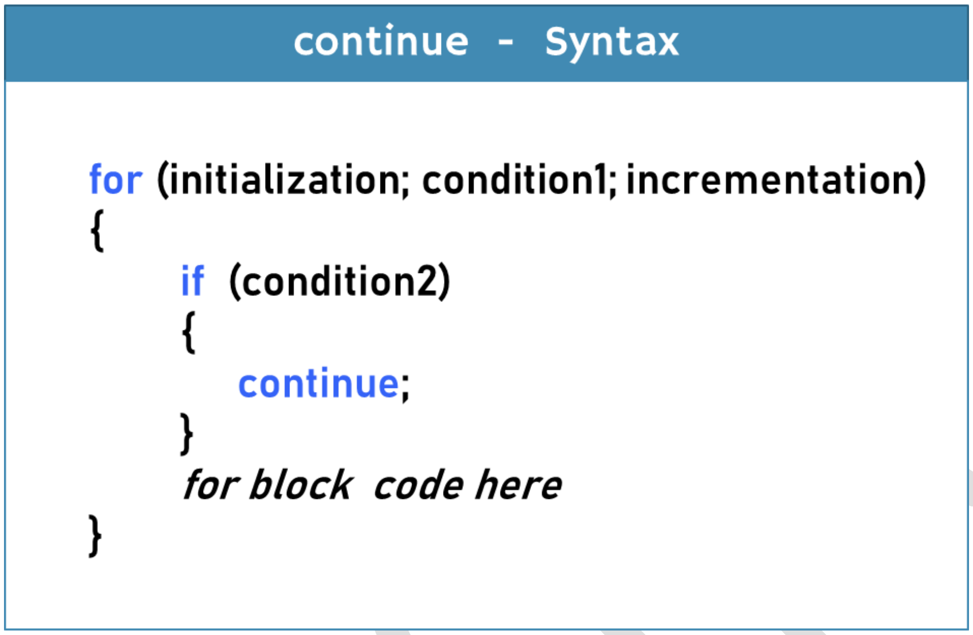


break - Example

```
for (int i = 0; i <= 10; i++)
{
    if (i == 6)
    {
        break;
    }
    System.Console.WriteLine(i);
}

//Output: 0, 1, 2, 3, 4, 5
```

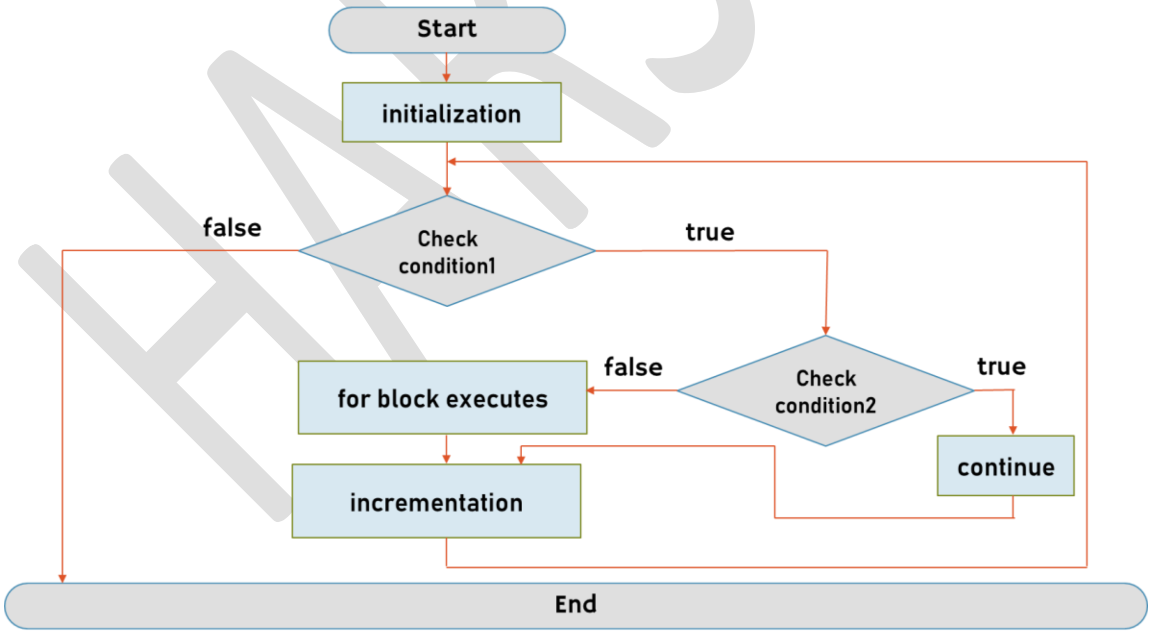
- Used to stop the execution of current loop.
- It is recommended to keep the "break" statement, inside "if" statement.
- It can be used in any type of loop (while, do-while, for).

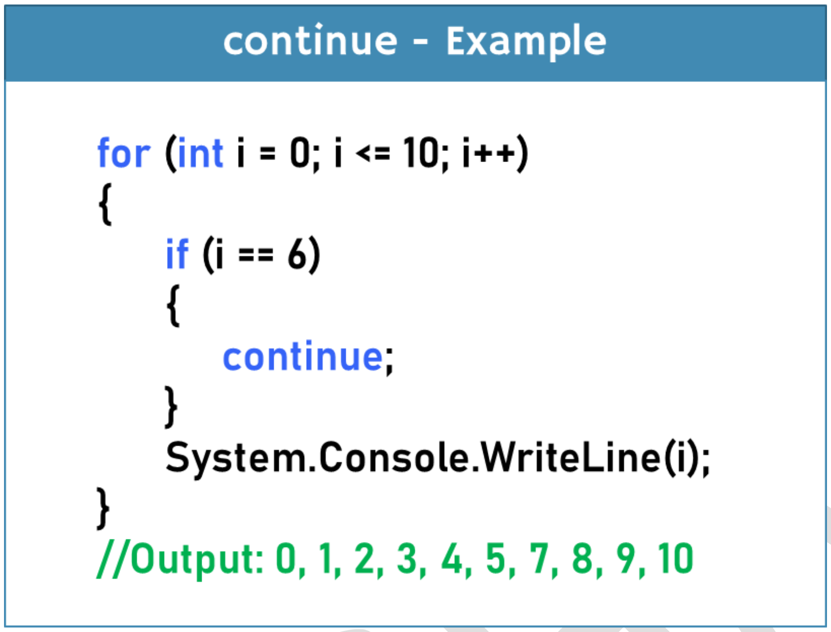
Continue



continue - Syntax

```
for (initialization; condition1; incrementation)
{
        if  (condition2)
        {
            continue;
        }
        for block  code here
}
```



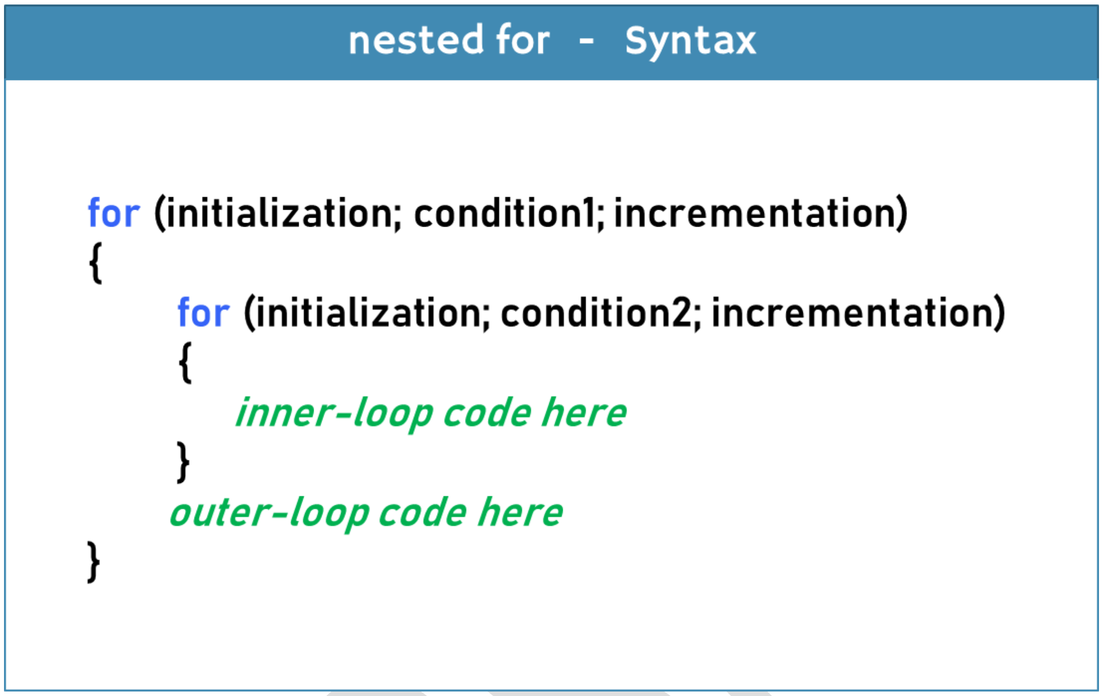


continue - Example

```
for (int i = 0; i <= 10; i++)
{
    if (i == 6)
    {
        continue;
    }
    System.Console.WriteLine(i);
}
//Output: 0, 1, 2, 3, 4, 5, 7, 8, 9, 10
```

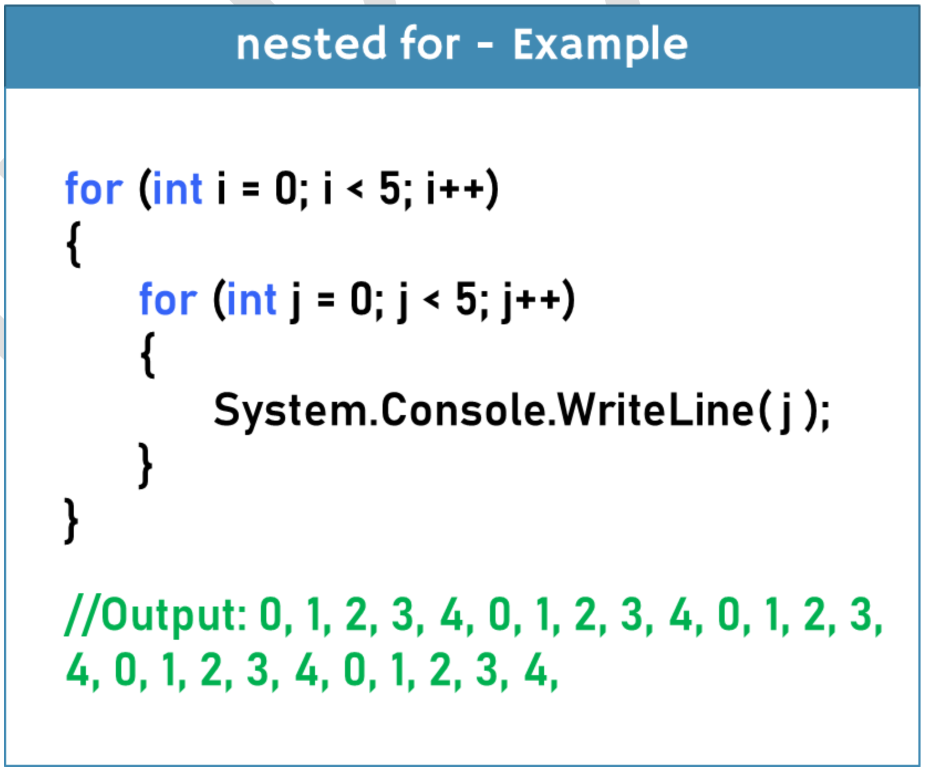
- Used to skip the execution of current iteration; and jump to the next iteration.
- It is recommended to keep the "continue" statement, inside "if" statement.
- It can be used in any type of loop (while, do-while, for).

Nested for



nested for - Syntax

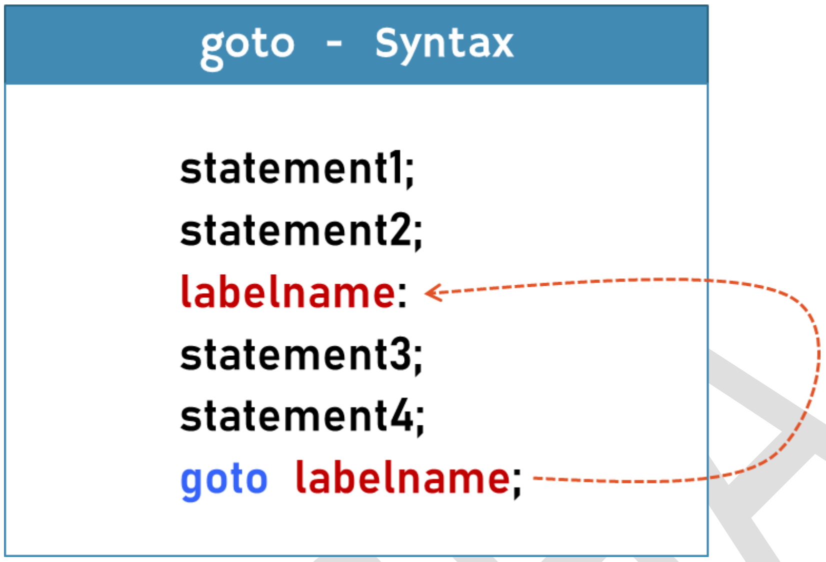
```
for (initialization; condition1; incrementation)
{
        for (initialization; condition2; incrementation)
        {
            inner-loop code here
        }
       outer-loop code here
}
```



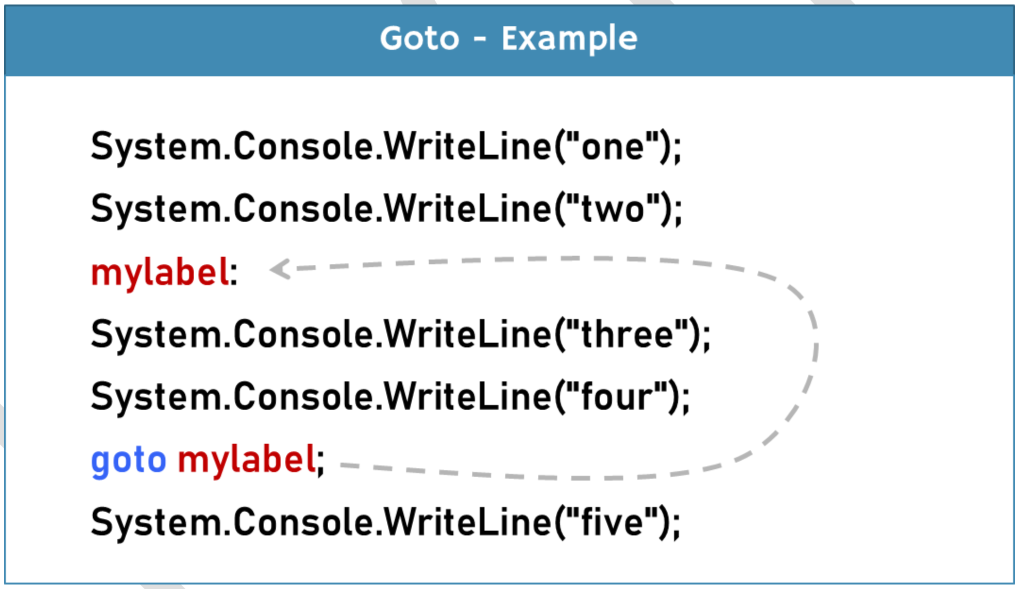
nested for - Example

```
for (int i = 0; i < 5; i++)
{
      for (int j = 0; j < 5; j++)
      {
            System.Console.WriteLine( j );
      }
}

//Output: 0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 0, 1, 2, 3,
4, 0, 1, 2, 3, 4, 0, 1, 2, 3, 4,
```

Goto



```
statement1;
statement2;
labelname:
statement3;
statement4;
goto labelname;
```



```
System.Console.WriteLine("one");
System.Console.WriteLine("two");
mylabel:
System.Console.WriteLine("three");
System.Console.WriteLine("four");
goto mylabel;
System.Console.WriteLine("five");
```

- Used to jump to the specific label.
- You must create a label with some specific name.
- The label can be present at the top of "goto statement"; or at the bottom; but it should be in the same method.