Technical Institute of Administration

Management Information System

Computer Programming

4. Flow control (if…else)

Lecturer:

Sipan M. Hameed

www.sipan.dev

2024-2025

1. Table of Contents

4. Boolean Expressions

A Boolean expression is any expression that evaluates to, or returns, a Boolean value.

```
// These expressions all evaluate to a boolean value.
// Therefore their values can be stored in boolean variables.
bool a = (2 > 1);
bool b = a && true;
bool c = !false || (7 < 8);
```

4.1 Boolean Type

The bool data type can be either true or false and is based on the concept that the validity of all logical statements must be either true or false.

```
bool skyIsBlue = true;
bool penguinsCanFly = false;
Console.WriteLine($"True or false, is the sky blue? {skyIsBlue}.");
```

Booleans encode the science of logic into computers, allowing for logical reasoning in programs. In a broad sense, the computer can encode the truthfulness or falseness of certain statements, and based on that information, completely alter the behavior of the program.

4.2 Logical Operators

Logical operators receive Boolean expressions as input and return a Boolean value. The && operator takes two Boolean expressions and returns true only if they both evaluate to true. The || operator takes two Boolean expressions and returns true if either one evaluates to true. The ! operator takes one Boolean expression and returns the opposite value.

```
// These variables equal true.
bool a = true && true;
bool b = false || true;
bool c = !false;

// These variables equal false.
bool d = true && false;
bool e = false || false;
bool f = !true;
```

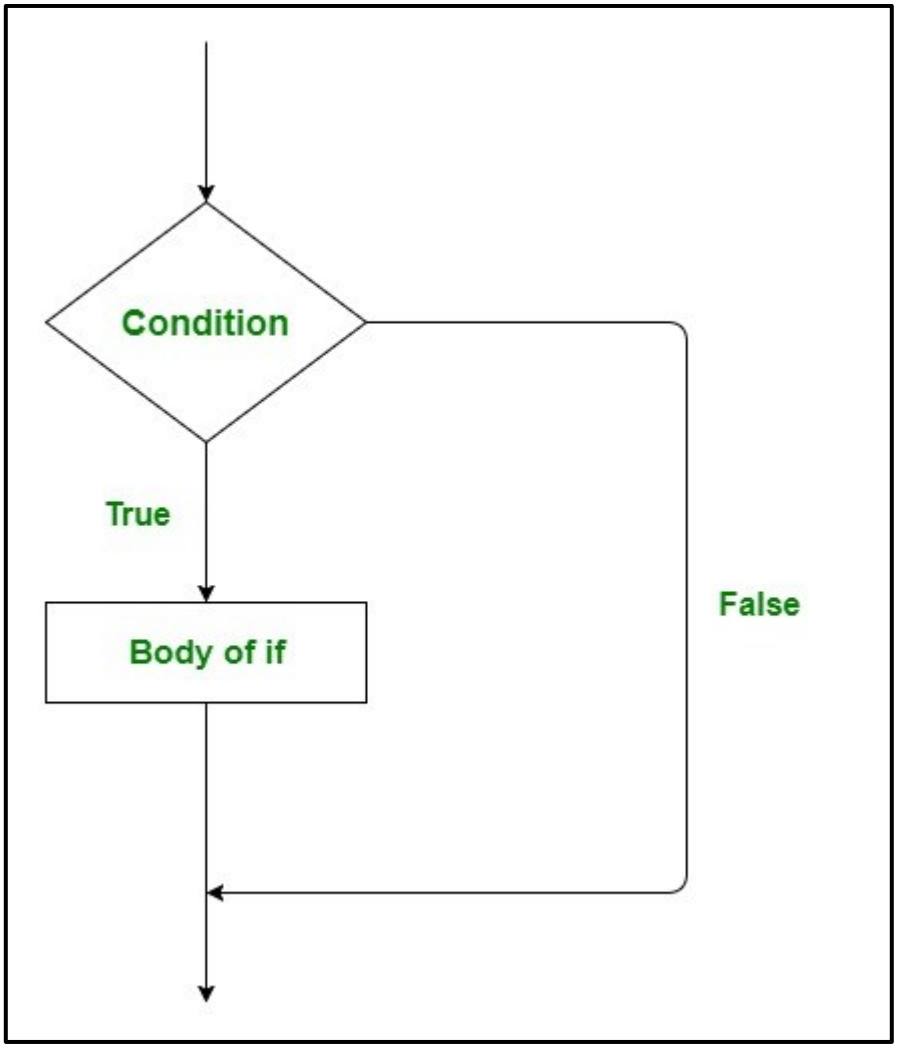
4.3 Flow control in C#

In C# language there are several keywords that are used to alter the flow of the program. When the program is run, the statements are executed from the top of the source file to the bottom. One by one. This flow can be altered by specific keywords. Statements can be executed multiple times. Some statements are called conditional statements. They are executed only if a specific condition is met.

4.4 C# if statement

The if statement has the following general form:

FlowChart:

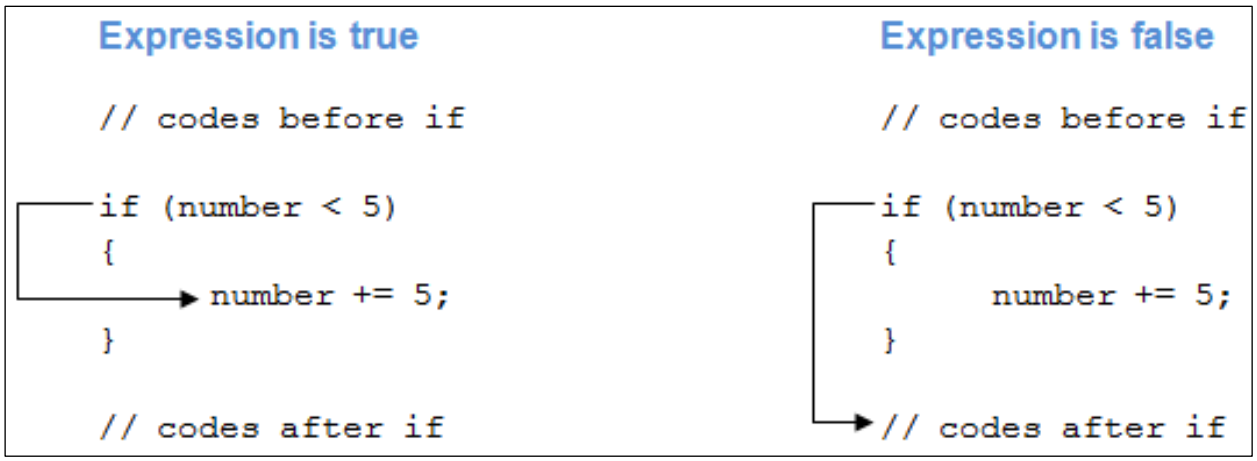



```
if (expression)
{
    statement;
}
```

The if keyword is used to check if an expression is true. If it is true, a statement is then executed. The statement can be a single statement or a compound statement. A compound statement consists of multiple statements enclosed by the block. A block is code enclosed by curly brackets.

```
using System;
namespace ex55
{
    class Program
    {
        static void Main()
        {
            int n = Convert.ToInt32(Console.ReadLine());

            if (n > 0)
            {
                Console.WriteLine("The n is positive");
            }

        }
    }
}
```

4.4.1 How if statement works



C# if-then statement will execute a block of code if the given condition is true. The syntax of if-then statement in C# is:

```
if (boolean-expression)
{
     // statements executed if boolean-expression is true
}
```

The boolean-expression will return either true or false.

If the boolean-expression returns true, the statements inside the body of if (inside {...}) will be executed.

If the boolean-expression returns false, the statements inside the body of if will be ignored.

For example,

```
if (number < 5)
{
      number += 5;
}
```

In this example, the statement

```
number += 5;
```

will be executed only if the value of number is less than 5.

Remember the += operator?

How if statement works?

How if statement works in C#?

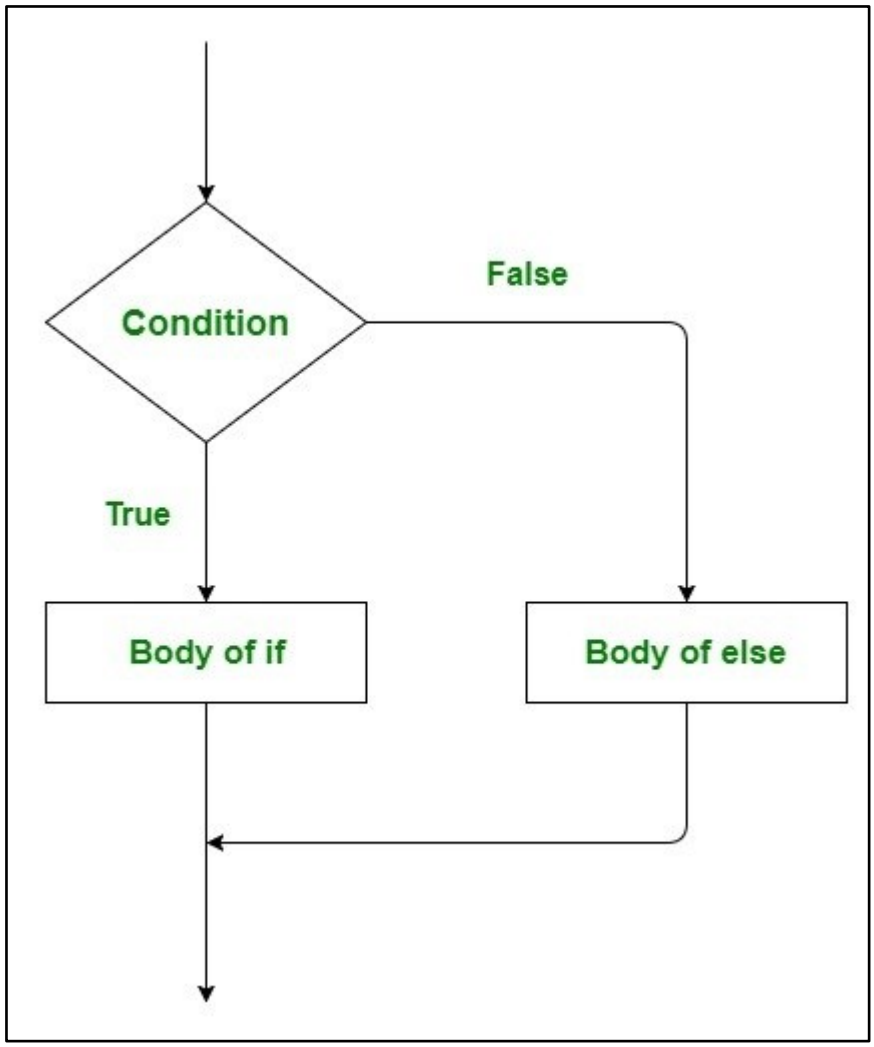
Example 1: C# if Statement

```
using System;
namespace Conditional
{
      class IfStatement
      {
            public static void Main()
            {
                  int number = 2;
                  if (number < 5)
                  {
                        Console.WriteLine("{0} is less than 5", number);
                  }

                  Console.WriteLine("This statement is always executed.");
            }
      }
}
```

When we run the program, the output will be:

```
2 is less than 5
This statement is always executed.
```

4.5 if...else (if-then-else) Statement



The if statement in C# may have an optional else statement. The block of code inside the else statement will be executed if the expression is evaluated to false.

The syntax of if...else statement in C# is:

```
if (boolean-expression)
{
     // statements executed if boolean-expression is true
}
else
{
     // statements executed if boolean-expression is false
}
```

For example,

```
if (number < 5)
{
     number += 5;
}
else
{
     number -= 5;
}
```

In this example, the statement

```
number += 5;
```

will be executed only if the value of number is less than 5.

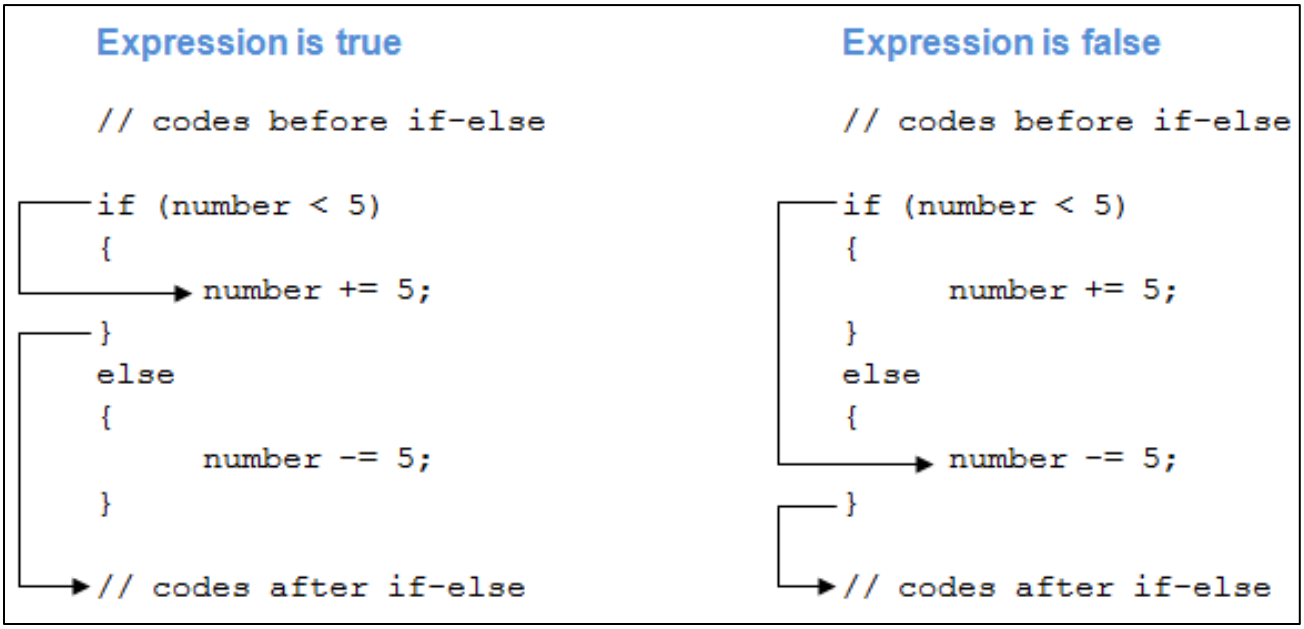
The statement

```
number -= 5;
```

will be executed if the value of number is greater than or equal to 5.

4.5.1 How if...else Statement works?

How if else statement works in C#?



Example 2: C# if...else Statement

```
using System;

namespace Conditional
{
       class IfElseStatement
       {
              public static void Main()
              {
                     int number = 12;

                     if (number < 5)
                     {
                            Console.WriteLine("{0} is less than 5", number);
                     }
                     else
                     {
                            Console.WriteLine("{0} is greater than or equal to 5", number);
                     }

                     Console.WriteLine("This statement is always executed.");
              }
       }
}
```

When we run the program, the output will be:

```
12 is greater than or equal to 5
This statement is always executed.
```

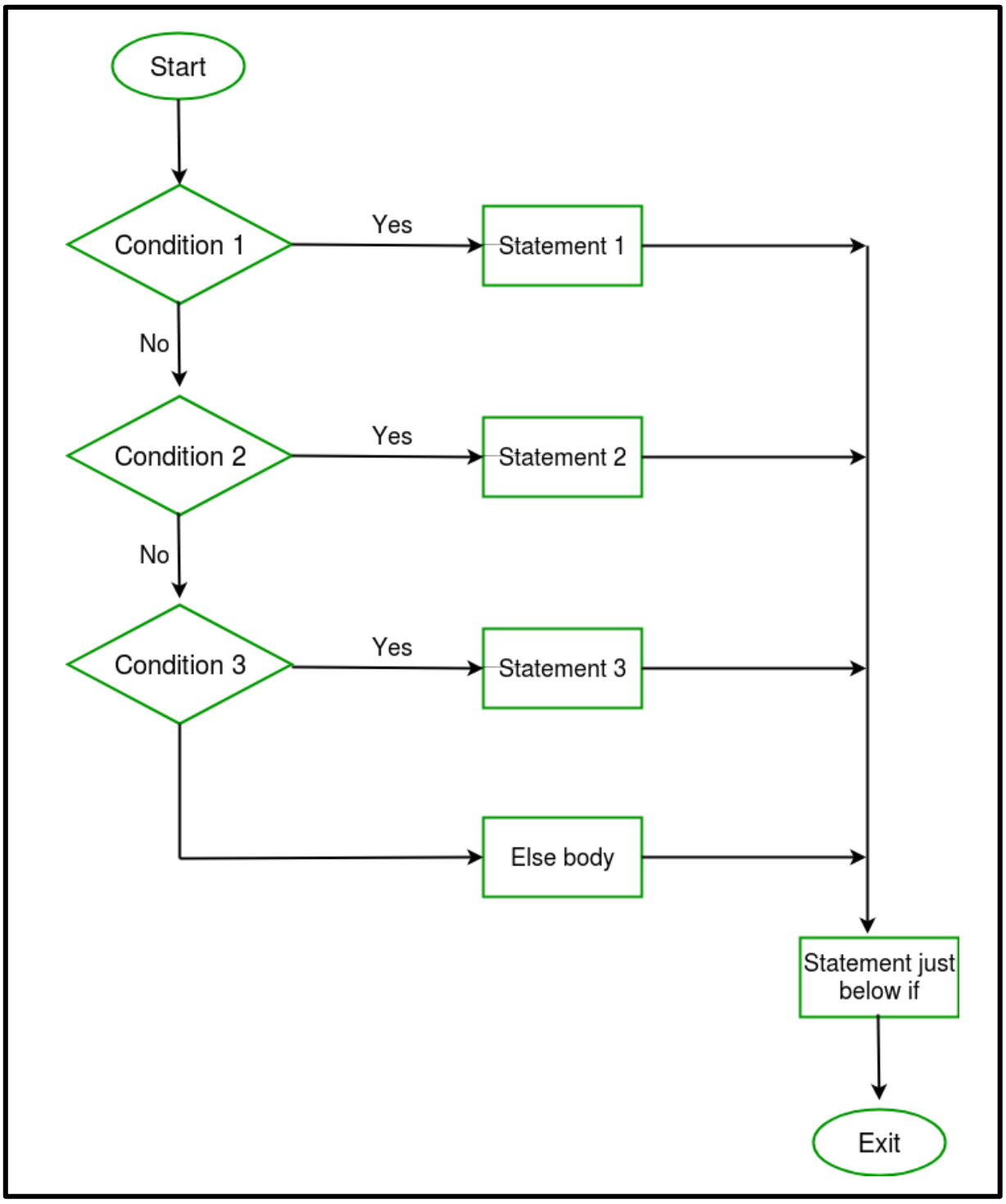
Here, the value of number is initialized to 12. So the expression number < 5 is evaluated to false. Hence, the code inside the else block are executed. The code after the if..else statement will always be executed irrespective to the expression.

Now, change the value of number to something less than 5, say 2. When we run the program the output will be:

```
2 is less than 5
This statement is always executed.
```

The expression `number < 5` will return true, hence the code inside if block will be executed.

4.6 C# if...else if (if-then-else if) Statement



When we have only one condition to test, if-then and if-then-else statement works fine. But what if we have a multiple condition to test and execute one of the many blocks of code.

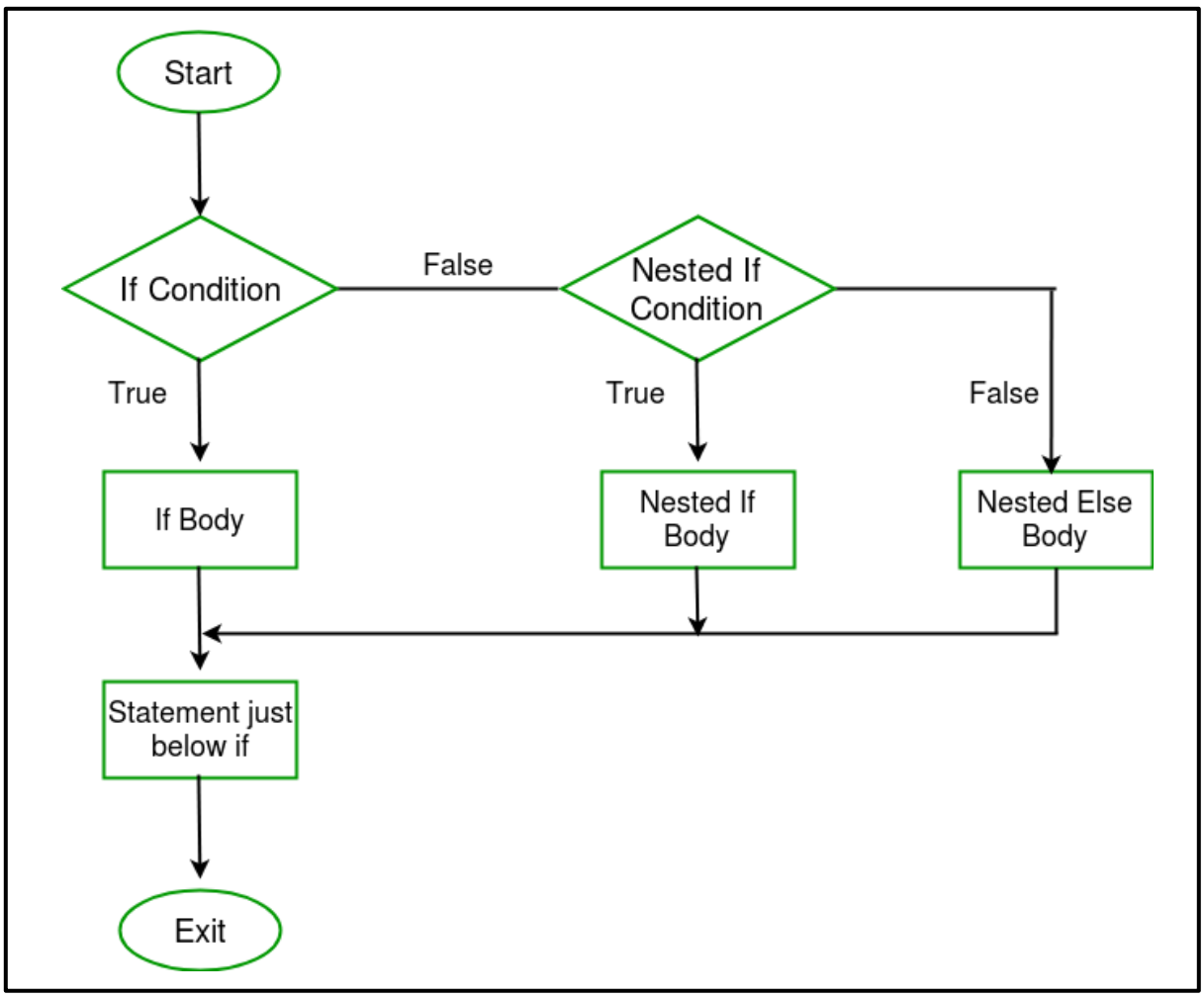
For such case, we can use if..else if statement in C#. The syntax for if...else if statement is:

```
if (boolean-expression-1)
{
     // statements executed if boolean-expression-1 is true
}
else if (boolean-expression-2)
{
     // statements executed if boolean-expression-2 is true
}
else if (boolean-expression-3)
{
     // statements executed if boolean-expression-3 is true
}
.
.
.
else
{
     // statements executed if all above expressions are false
}
```

The if...else if statement is executed from the top to bottom. As soon as a test expression is true, the code inside of that if (or else if) block is executed. Then the control jumps out of the if...else if block.

If none of the expression is true, the code inside the else block is executed.

4.7 Nested if...else Statement



An if...else statement can exist within another if...else statement. Such statements are called nested if...else statement.

The general structure of nested if…else statement is:

```
if (boolean-expression)
{
     if (nested-expression-1)
     {
          // code to be executed
     }
     else
     {
     // code to be executed
     }
}
else
{
     if (nested-expression-2)
     {
          // code to be executed
     }
     else
     {
          // code to be executed
     }
}
```

Nested if statements are generally used when we have to test one condition followed by another. In a nested if statement, if the outer if statement returns true, it enters the body to check the inner if statement.

4.8 Summary: Remember: -

4.8.1 Summery Data Types

Data Type	Represents	Memory Size	Range	Default Value
Bool	Boolean Value	1 byte	True or False	False
Char	Unicode character	1 byte	U +0000 to U +ffff	'\0'
int	Signed integer type	4 byte	-2,147,483,648 to 2,147,483,647	0
double	Double-precision floating point type	8 byte	(+/-)5.0 x 10-324 to (+/-)1.7 x 10308	0.0D
string	Stores a sequence of characters, surrounded by double quotes	2 bytes per character		null

4.8.2 Summary operations: -

Operator Name	Operator	Examples (Output value)
Arithmetic Operators	+ , - , * , / , % , ++ , --	45, 3.18
Assignment Operators	= , += , -= , *= , /=	x = 5
Comparison Operators	== , != , > , < , >= , <=	True, False
Logical Operators	&& , || , !	x > 5 && x < 10

4.9 Practical Examples

4.9.1 Example: Positive or negative number

Check whether number Positive or negative:

Code:

```
using System;

public class ex41
{
    static void Main()
    {

        double x = Convert.ToDouble(Console.ReadLine());

        if (x>0)
        {
            Console.WriteLine("positive number");
        }
        else
        {
            Console.WriteLine("negative number");
        }

    }
}
```

Output:

```
58
positive number
```

4.9.2 Example: odd and even number

Check whether number is odd or even

Code:

```
using System;

public class ex42
{
    static void Main()
    {

        double x = Convert.ToDouble(Console.ReadLine());

        if (x%2==0)
        {
            Console.WriteLine("even number");
        }
        else
        {
            Console.WriteLine("odd number");
        }

    }
}
```

Output:

```
43
odd number
```

4.9.3 Example:

Check a number for the following case:

If number between 100-90 print A.

If number between 89-80 print B

If number between 79-70 print F

Code:

```
using System;

public class ex43
{
    static void Main()
    {

        double x = Convert.ToDouble(Console.ReadLine());

        if (x<=10 && x>=90)
        {
            Console.WriteLine("A");
        }
        if (x<=89 && x>=90)
        {
            Console.WriteLine("B");
        }
        if (x<=79 && x>=70)
        {
            Console.WriteLine("F");
        }

    }
}
```

Output:

```
85
B
```

4.9.4 Example: increment and decrement

Enter a number and check If the number is greater than 60 increment by 10, else decrement by 25

Code:

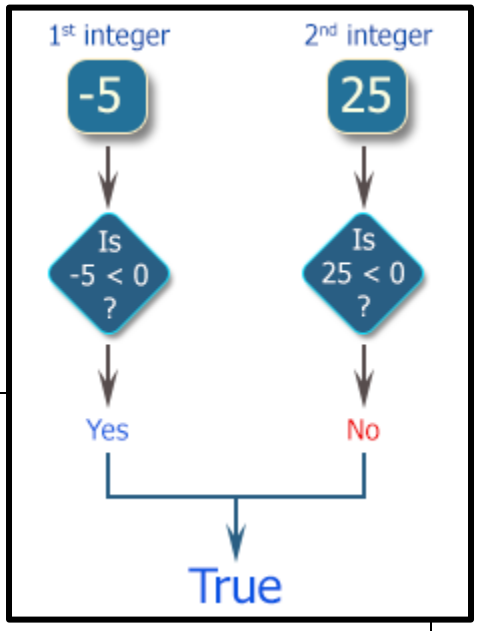
```
using System;

public class ex44
{
    static void Main()
    {

        double x = Convert.ToDouble(Console.ReadLine());

        if (x>60)
        {
            x += 10;
        }
        else
        {
            x -= 25;
        }

        Console.WriteLine(x);
    }
}
```

Output:

```
77
87
```

4.9.5 Example:

Write a C# program to check two given integers and return true if one is negative and one is positive.



Solution:

C# Sharp Code:

```
using System;
public class ex45
{
  static void Main()
        {
           Console.WriteLine("\nInput first integer:");
           int x = Convert.ToInt32(Console.ReadLine());
           Console.WriteLine("Input second integer:");
           int y = Convert.ToInt32(Console.ReadLine());
           Console.WriteLine("Check if one is negative and one is positive:");
           Console.WriteLine((x < 0 && y > 0) || (x > 0 && y < 0));
        }
}
```

Sample Output:

```
Input first integer:
-5
Input second integer:
25
Check if one is negative and one is positive:
True
```