UEH UNIVERSITY | College of Technology and Design

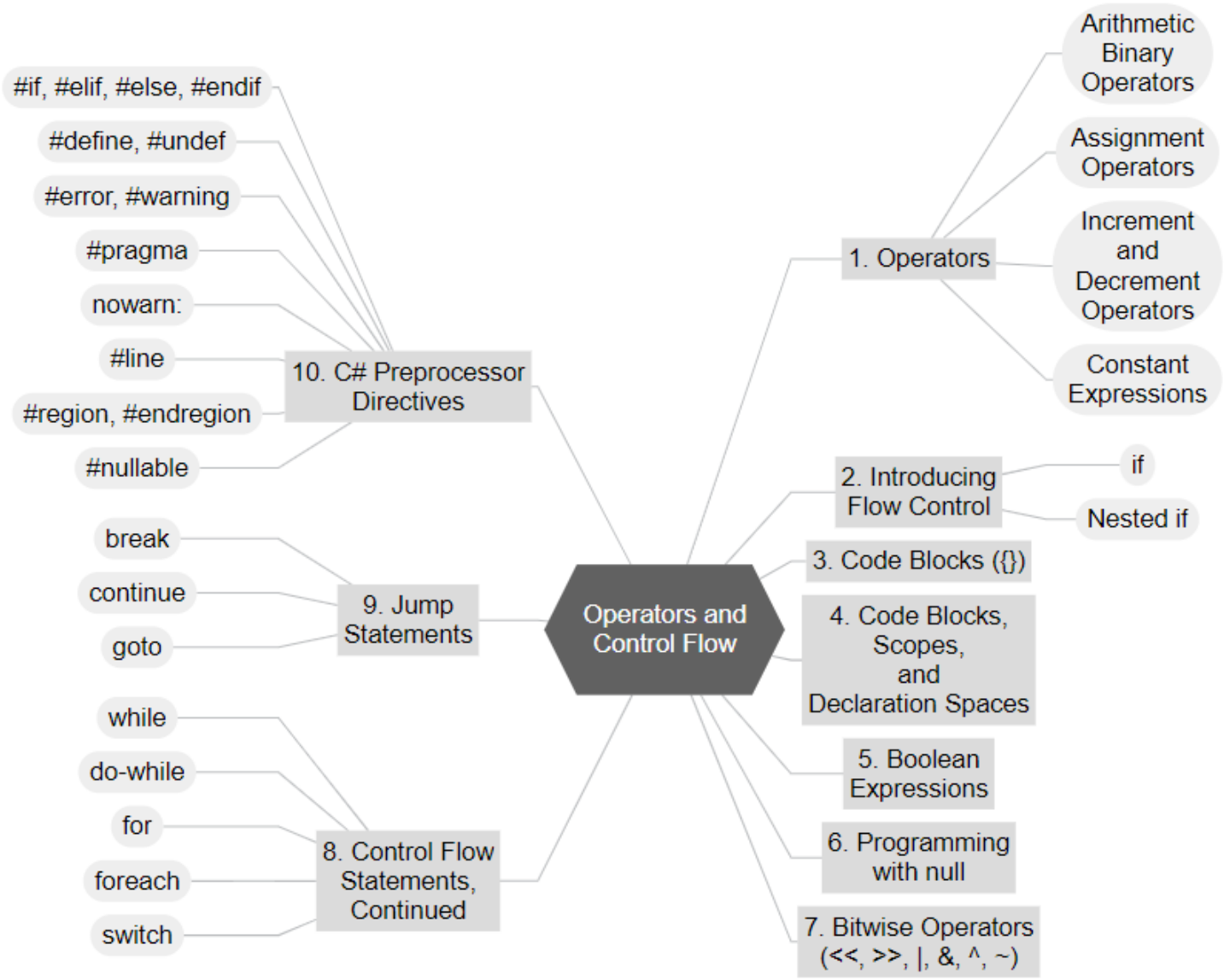
Fundamentals of Programming

Operators and Control Flow

By: Võ Văn Hải

Email: vovanhai@ueh.edu.vn

Objectives


Operators and Control Flow
1. Operators
Arithmetic Binary Operators
Assignment Operators
Increment and Decrement Operators
Constant Expressions
2. Introducing Flow Control
if
Nested if
3. Code Blocks ({})
4. Code Blocks, Scopes, and Declaration Spaces
5. Boolean Expressions
6. Programming with null
7. Bitwise Operators (<<, >>, |, &, ^, ~)
8. Control Flow Statements, Continued
while
do-while
for
foreach
switch
9. Jump Statements
break
continue
goto
10. C# Preprocessor Directives
#if, #elif, #else, #endif
#define, #undef
#error, #warning
#pragma
nowarn:
#line
#region, #endregion
#nullable

Operators

Introduction

- Operators are used to perform mathematical or logical operations on values (or variables) called operands to produce a new value called the result.
- There are three operator categories—unary, binary, and ternary—corresponding to the number of operands (one, two, and three, respectively).
  - Furthermore, while some operators are represented with symbols like +, -, ?., and ??, other operators take the form of keywords, like default and is.
- Operators in C# :
  - Unary – take one operand
  - Binary – take two operands
  - Ternary (?:) – takes three operands

Operators

Categories of Operators in C#

Category	Operators
Arithmetic	+ - * / %
Logical	&& || ^ !
Binary	& | ^ ~ << >>
Comparison	== != < > <= >=
Assignment	= += -= *= /= %= &= |= ^= <<= >>=
String concatenation	+
Type conversion	is as typeof
Other	. [] () ?: new

Operators

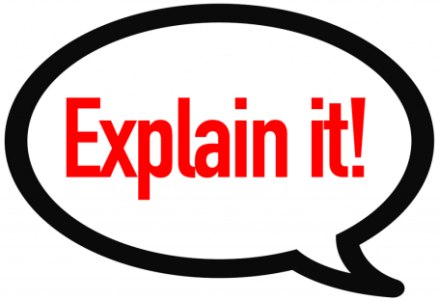
Operators Precedence

Precedence	Operators
Highest	()
	++ -- (postfix) new typeof
	++ -- (prefix) + - (unary) ! ~
	* / %
	+ -
	<< >>
	< > <= >= is as
	== !=&
	^
	|
	&&
	||
	?:
Lowest	= *= /= %= += -= <<= >>= &= ^= |=

- Parenthesis operator always has highest precedence
- **Note**: prefer using parentheses, even when it seems stupid to do so

Operators

Example (1)

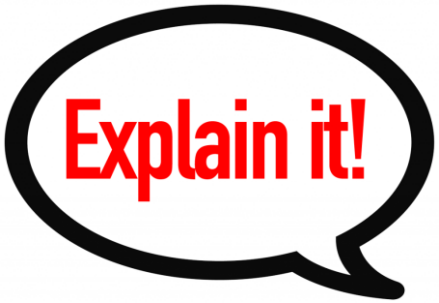



```
int squarePerimeter = 17;
double squareSide = squarePerimeter / 4.0;
double squareArea = squareSide * squareSide;
Console.WriteLine(squareSide); // 4.25
Console.WriteLine(squareArea); // 18.0625

int a = 5, b = 4;
Console.WriteLine(a + b); // 9
Console.WriteLine(a + b++); // 9
Console.WriteLine(a + b); // 10
Console.WriteLine(a + (++b)); // 11
Console.WriteLine(a + b); // 11
```

Operators

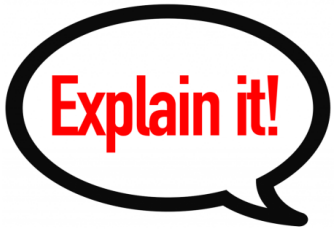
Example (2)




```
Console.WriteLine(12 / 3); // 4
Console.WriteLine(11 / 3); // 3
Console.WriteLine(11.0 / 3); //
Console.WriteLine(11 / 3.0); //
Console.WriteLine(11 % 3); // 2
Console.WriteLine(11 % -3); // 2
Console.WriteLine(-11 % 3); // -2
Console.WriteLine(1.5 / 0.0); // Infinity
Console.WriteLine(-1.5 / 0.0); // -Infinity
Console.WriteLine(0.0 / 0.0); // NaN
int x = 0;
Console.WriteLine(5 / x); // DivideByZeroException
```

Operators

Example (3)




```
int bigNum = 2000000000;
int bigSum = 2 * bigNum; // Integer overflow!
Console.WriteLine(bigSum); //
bigNum = Int32.MaxValue;
bigNum = bigNum + 1;
Console.WriteLine(bigNum); //
checked
{
    // This will cause OverflowException
    bigSum = bigNum * 2;
}
```

Operators

Exercises

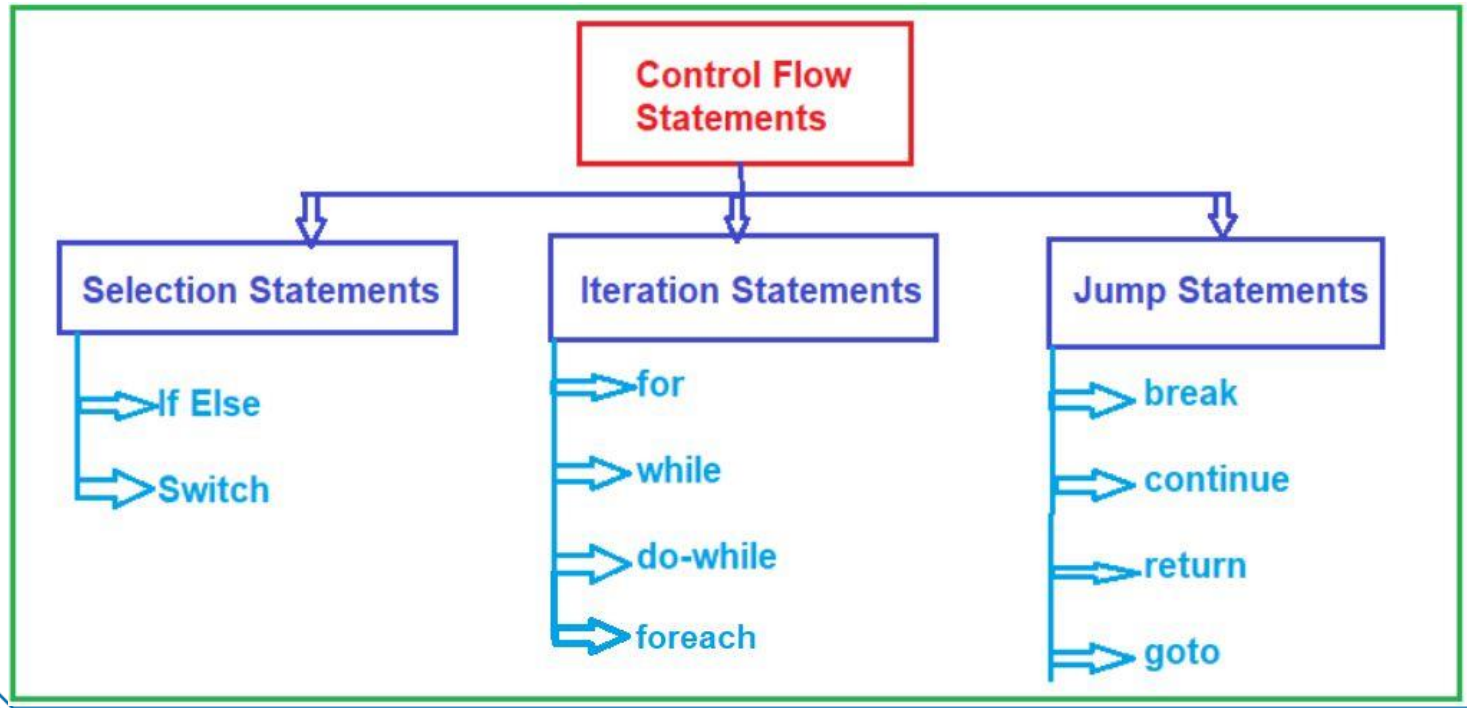
1. Write a C# Sharp program that takes two numbers as input and performs an operation (+,-,*,x,/) on them and displays the result of that operation.
2. Write a C# Sharp program to display certain values of the function x = y2 + 2y + 1 (using integer numbers for y, ranging from -5 to +5).
3. Write a C# Sharp program that takes distance and time (hours, minutes, seconds) as input and displays speed in kilometers per hour (km/h) and miles per hour (miles/h).
4. Write a C# Sharp program that takes the radius of a sphere as input and calculates and displays the surface and volume of the sphere. $V = 4/3*\pi*r^3$
5. Write a C# Sharp program that takes a character as input and checks if it is a vowel, a digit, or any other symbol.

Control Flow Statements

Control Flow Statements

Introduction

- The Control Flow Statements are the statements that Alter the Flow of Program Execution and provide better control to the programmer on the flow of execution.
- The Control Flow Statements are useful to write better and more complex programs. A program executes from top to bottom except when we use control statements.
- We can control the order of execution of the program, based on logic and values.



Types of Control Flow Statements in C#:

Control flow statements

If statement

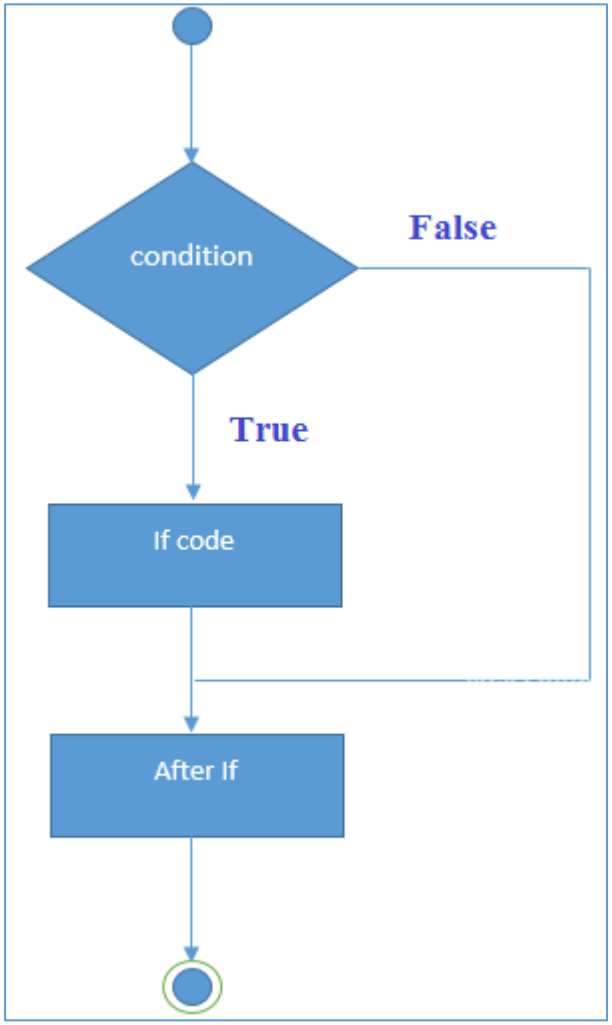
Statement	General Syntax Structure	Example

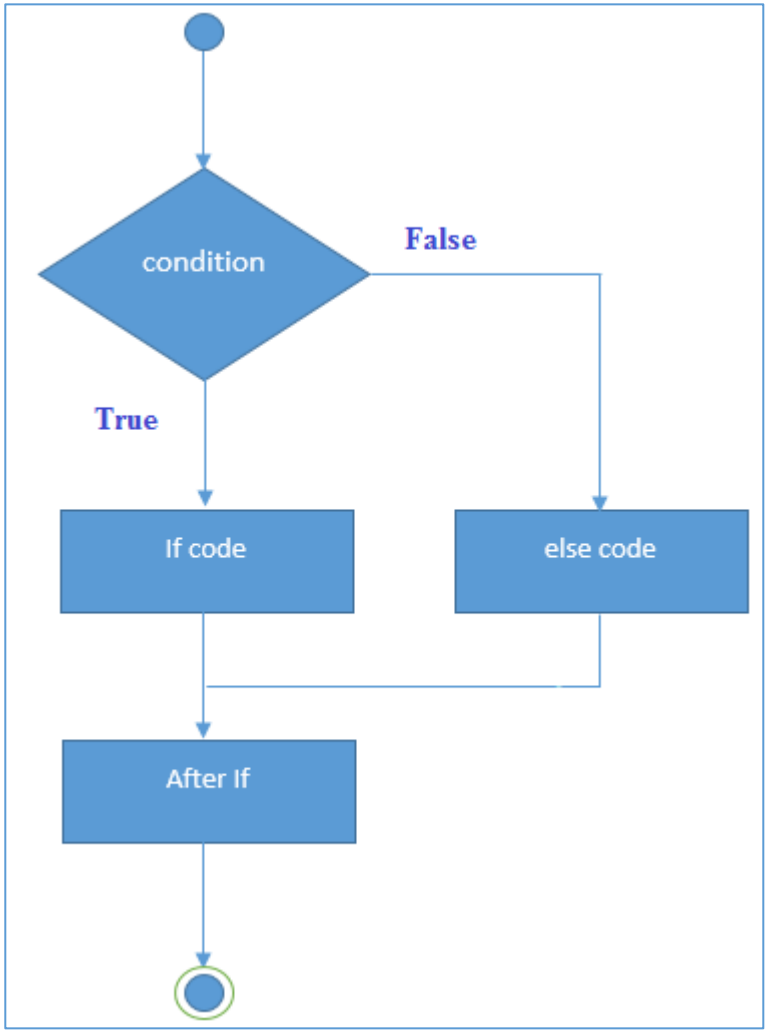
`if` statement	`if(`*boolean-expression*`)`
	  embedded-statement
		`if (input == "quit")`
		`{`
		  `Console.WriteLine(`
		  `"Game end");`
		  `return;`
		`}`

	`if(`*boolean-expression*`)`
	  embedded-statement
	`else`
	  embedded-statement
		`if (input == "quit")`
		`{`
		  `Console.WriteLine(`
		  `"Game end");`
		  `return;`
		`}`
		`else`
		  `GetNextMove();`

Control flow statements

If statement diagram





Control flow statements

While and do…while statement

while statement	`while(boolean-expression)`
	
	`  embedded-statement`
		`while(count < total)`
		`{`
		`  Console.WriteLine(`
		`  $"count = {count}");`
		`  count++;`
		`}`

do while statement	`do`
	
	`  embedded-statement`
	
	`while(boolean-expression);`
		`do`
		`{`
		`  Console.WriteLine(`
		`  "Enter name:");`
		`  input =`
		`  Console.ReadLine();`
		`}`
		`while(input != "exit");`

Control flow statements

For statement

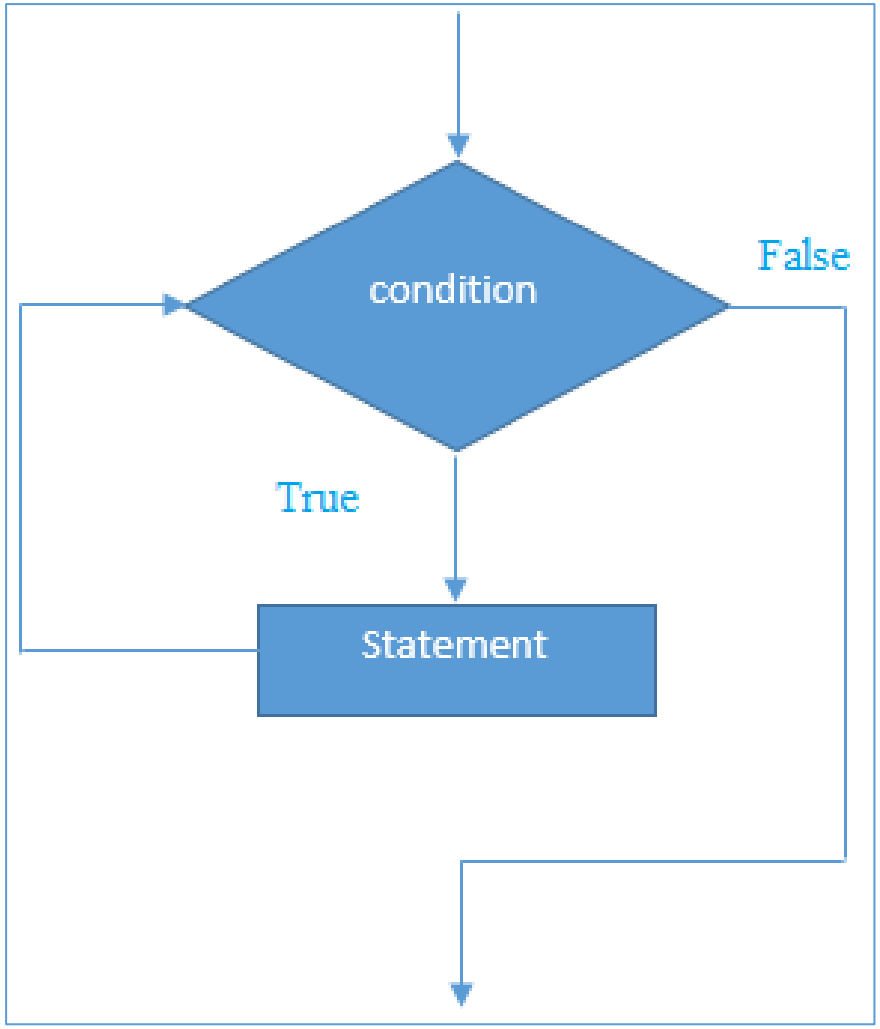
for statement	`for(`*for-initializer;*
	boolean-expression;
	for-iterator)
	embedded-statement
		`for (int count = 1;`
		`count <= 10;`
		`count++)`
		`{`
		`Console.WriteLine(`
		`$"count = {count}");`
		`}`

foreach statement	`foreach(`*type identifier* `in`
	expression)
	embedded-statement
		`foreach (char letter in email)`
		`{`
		`if(!insideDomain)`
		`{`
		`if (letter == '@')`
		`{`
		`insideDomain = true;`
		`}`
		`continue;`
		`}`
		`Console.Write(letter);`
		`}`

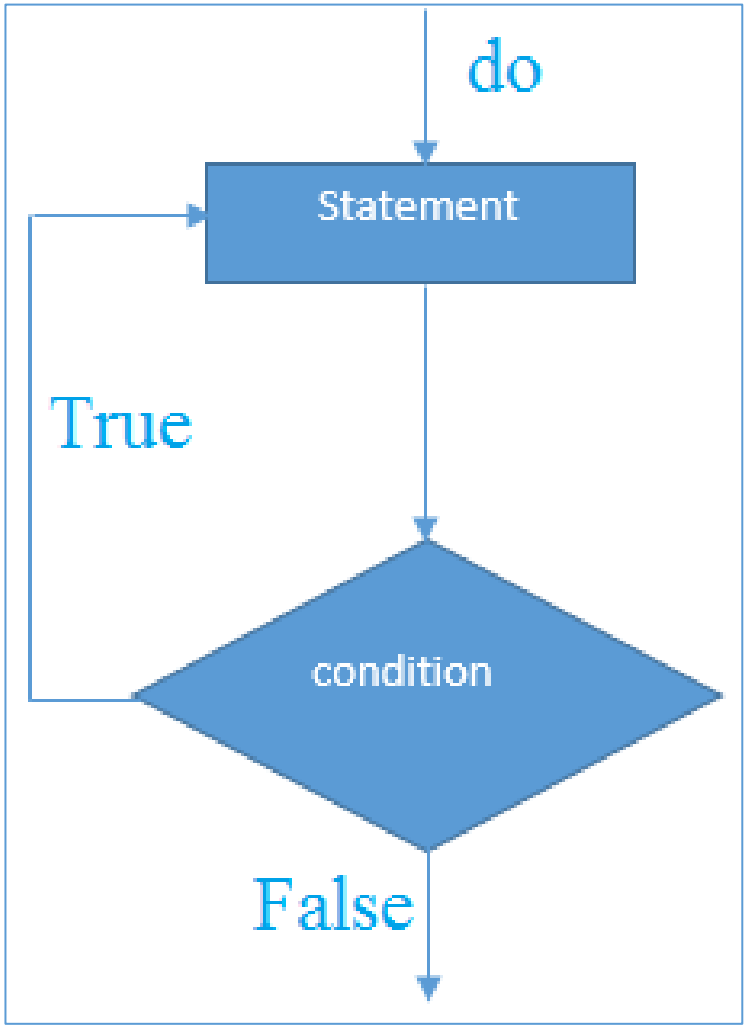
Control flow statements

while and do…while diagrams

while statement

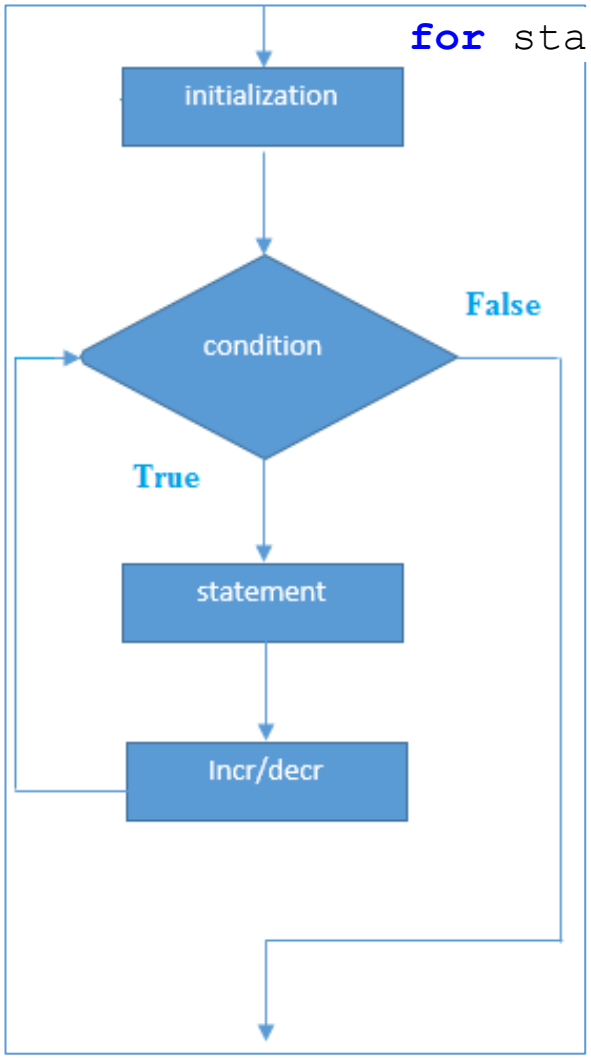


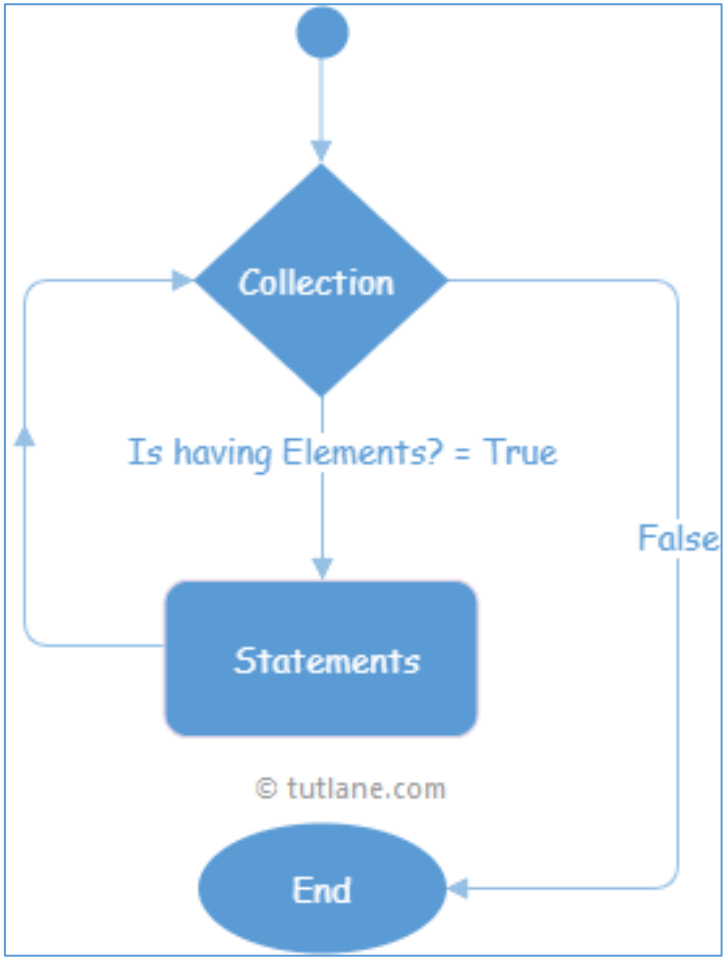
do while statement



Control flow statements

for and foreach diagrams





Control flow statements

Switch and continue statements

`continue` statement	`continue;`	

`switch` statement	`switch(governing-type-`
	`expression)`
	`{`
	`  ...`
	`  case const-expression:`
	`  statement-list`
	`  jump-statement`
	`  default:`
	`  statement-list`
	`  jump-statement`
	`}`
		`switch(input)`
		`{`
		`  case "exit":`
		`  case "quit":`
		`  Console.WriteLine(`
		`  "Exiting app....");`
		`  break;`
		`  case "restart":`
		`  Reset();`
		`  goto case "start";`
		`  case "start":`
		`  GetNextMove();`
		`  break;`
		`  default:`
		`  Console.WriteLine(`
		`  input);`
		`  break;`
		`}`

Control flow statements

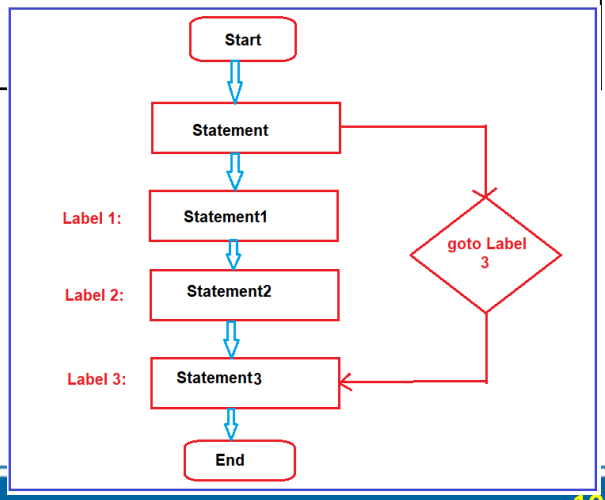
Break and goto statements

`break` statement	`break;`	`for (int i = 1; i < 4; i++) {`
		`  if (i == 3) break;`
		`  Console.WriteLine(i);`
		`}`

`goto` statement	`goto identifier;`	`switch (number){`
		`  case 5: Console.WriteLine("case 5");`
		`  break;`
		`  case 10: Console.WriteLine("case 10");`
		`  break;`
		`  case 20: Console.WriteLine("case 20");`
		`  // goto statement transfer the control to case 5`
		`  goto case 5;`
		`  default: Console.WriteLine("No match found");`
		`  break;`
		`}`

	`goto case const-expression;`	
	`goto default;`	

```
public static void Main(string[] args) {
    for (int i = 0; i < 10; i++) {
        Console.WriteLine(i);
        if (i == 5) goto End_case;
    }
End_case:
    Console.WriteLine("End case, exit"); return;
}
```



Control flow statements

Break and goto statements

```
int i = 1;
while (i < 10)
{
    //Statements
    if(i == 5)
    {
        break;
    }
    //Statements
}

//Statements
```

Using break in While Loop

```
int i = 1;
do
{
    //Statements
    if (i == 5)
    {
        break;
    }
    //Statements
} while (i > 10);
//Statements
```

Using break in Do While Loop

```
for (int i = 0; i < 10; i++)
{
    //Statements
    if (i == 5)
    {
        break;
    }
    //Statements
}
//Statements
```

Using break in for loop

```
int i = 1;
while(i <= 5)
{
    //Statements
    if(i == 3)
    {
        continue;
    }
    //Statements
}
```

Using continue in While Loop

```
int i = 1;
do
{
    //Statements
    if (i == 3)
    {
        continue;
    }
    //Statements
}
while (i <= 5);
```

Using continue in do While Loop

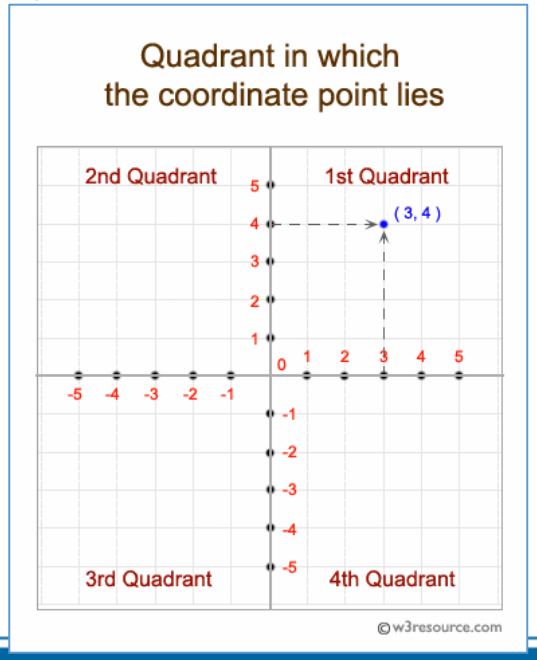
```
for (int i = 1; i < 5; i++)
{
    //Statements
    if (i == 3)
    {
        continue;
    }
    //Statements
}
```

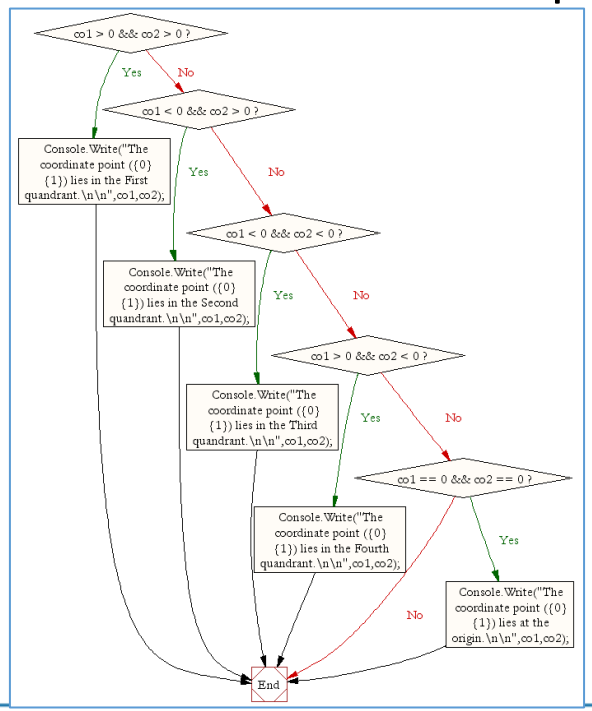
Using continue in for Loop

Control flow statements

Exercises

1. Write a C# Sharp program to check whether a given number is even or odd.
2. Write a C# Sharp program to find the largest of three numbers.
3. Write a C# Sharp program to accept a coordinate point in an XY coordinate system and determine in which quadrant the coordinate point lies.





Test Data :
Input the value for X coordinate :7
Input the value for Y coordinate :9
Expected Output :
The coordinate point (7,9) lies in the First quadrant.

Control flow statements

Exercises

1. Write a program to check whether a triangle is Equilateral, Isosceles or Scalene.
2. Write a program to read 10 numbers and find their average and sum.
3. Write a program to display the multiplication table of a given integer.
4. Write a program to display a pattern like triangles with a number.
5. The patterns like :

```
1
12
123
1234
```

```
1
2 3
4 5 6
7 8 9 10
```

```
   1
  2 3
 4 5 6
7 8 9 10
```

6. Write a program to display the n terms of harmonic series and their sum. 1 + 1/2 + 1/3 + 1/4 + 1/5 ... 1/n terms
7. Write a program to find the ‘perfect’ numbers within a given number range.
8. Write a program to determine whether a given number is prime or not.

Thank you
for Listening!