مقدمة

Primitive Data Type

Data Types	Default Value	Minimum Value	Maximum Value
sbyte	0	-128	127
byte	0	0	255
short	0	-32768	32767
ushort	0	0	65535
int	0	-2147483648	2147483647
uint	0u	0	4294967295
long	0L	-9223372036854775808	9223372036854775807
ulong	0u	0	18446744073709551615

Primitive Data Type

Data Types	Default Value	Minimum Value	Maximum Value
float	0.0f	$\pm 1.5 \times 10^{-45}$	$\pm 3.4 \times 10^{38}$
double	0.0d	$\pm 5.0 \times 10^{-324}$	$\pm 1.7 \times 10^{308}$
decimal	0.0m	$\pm 1.0 \times 10^{-28}$	$\pm 7.9 \times 10^{28}$
bool	false	Two possible values: true and false	
char	'\u0000'	'\u0000'	'\uffff'
object	null	-	-
string	null	-	-

Operator Categories

- Below is a list of the operators, separated into categories:

Category	Operators
arithmetic	-, +, *, /, %, ++, --
logical	&&, ||, !, ^
binary	&, |, ^, ~, <<, >>
comparison	==,!=, >, <, >=, <=
assignment	=, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=
string concatenation	+
type conversion	(type), as, is, typeof, sizeof
other	., new, (), [], ?:, ??

Types of Operators by Number of Arguments

- Operators can be separated into different types according to the number of arguments they could take:

Operator type	Number of arguments (operands)
unary	takes one operand
binary	takes two operands
ternary	takes three operands

Types of Operators by Number of Arguments

```
int a = 5;
int 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
Console.WriteLine(14 / a);      // 2
Console.WriteLine(14 % a);      // 4
```

Logical Operators

x	y	!x	x && y	x || y	x ^ y
true	true	false	true	true	false
true	false	false	false	true	true
false	true	true	false	true	true
false	false	true	false	false	false

Logical Operators

```
bool a = true;
bool b = false;
Console.WriteLine(a && b);                  // False
Console.WriteLine(a || b);                  // True
Console.WriteLine(!b);                      // True
Console.WriteLine(b || true);               // True
Console.WriteLine((5 > 7) ^ (a == b));      // False
```

Bitwise Operators

x	y	~x	x & y	x | y	x ^ y
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0

Bitwise Operators

```
byte a = 3;                     // 0000 0011 = 3
byte b = 5;                     // 0000 0101 = 5

Console.WriteLine(a | b);   // 0000 0111 = 7
Console.WriteLine(a & b);   // 0000 0001 = 1
Console.WriteLine(a ^ b);   // 0000 0110 = 6
Console.WriteLine(~a & b);  // 0000 0100 = 4
Console.WriteLine(a << 1);  // 0000 0110 = 6
Console.WriteLine(a << 2);  // 0000 1100 = 12
Console.WriteLine(a >> 1);  // 0000 0001 = 1
```

Type Conversion and Casting Generally, operators

- - implicit conversion;
- - explicit conversion;
- - conversion to or from **string**;

Type Conversion and Casting Generally, operators

- **Possible Implicit Conversions**
- Here are some possible implicit conversions of primitive data types in C#:
- - **sbyte → short, int, long, float, double, decimal;**
- - **byte → short, ushort, int, uint, long, ulong, float, double, decimal;**
- - **short → int, long, float, double, decimal;**
- - **ushort → int, uint, long, ulong, float, double, decimal;**
- - **char → ushort, int, uint, long, ulong, float, double, decimal**

Type Conversion and Casting Generally, operators

- **Possible Implicit Conversions**
- Here are some possible implicit conversions of primitive data types in C#:
- - **sbyte → short, int, long, float, double, decimal;**
- - **byte → short, ushort, int, uint, long, ulong, float, double, decimal;**
- - **short → int, long, float, double, decimal;**
- - **ushort → int, uint, long, ulong, float, double, decimal;**
- - **char → ushort, int, uint, long, ulong, float, double, decimal**

Type Conversion and Casting
Generally, operators

- - **uint → long, ulong, float, double, decimal;**
- - **int → long, float, double, decimal;**
- - **long → float, double, decimal;**
- - **ulong → float, double, decimal;**
- - **float → double.**

Type Conversion and Casting Generally, operators

- **Explicit Type Conversion**

```
double myDouble = 5.1d;
Console.WriteLine(myDouble); // 5.1

long myLong = (long)myDouble;
Console.WriteLine(myLong); // 5

myDouble = 5e9d; // 5 * 10^9
Console.WriteLine(myDouble); // 5000000000

int myInt = (int)myDouble;
Console.WriteLine(myInt); // -2147483648
Console.WriteLine(int.MinValue); // -2147483648
```

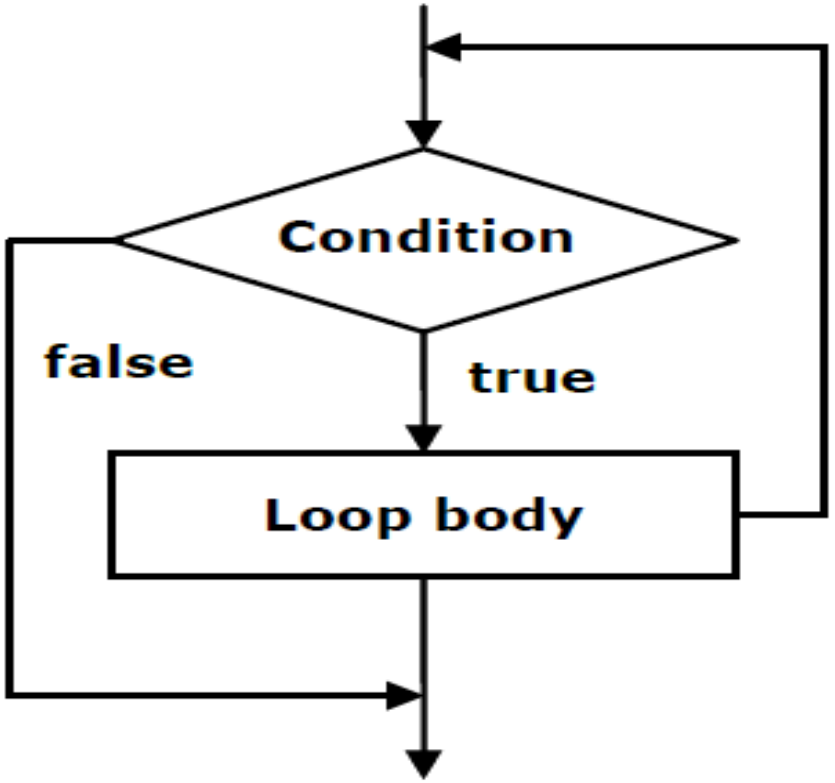
If-else condition

```
static void Main()
{
   int x = 2;
   if (x > 3)
   {
      Console.WriteLine("x is greater than 3");
   }
   else
   {
      Console.WriteLine("x is not greater than 3");
   }
}
```

If-else if -else condition

```
int x = 5;
if(x > 0)
{
        Console.WriteLine("positive");
}
else if (x < 0)
{
        Console.WriteLine("negitive");
}
else
{
        Console.WriteLine("is zero");
}
```

Loop (while loop)



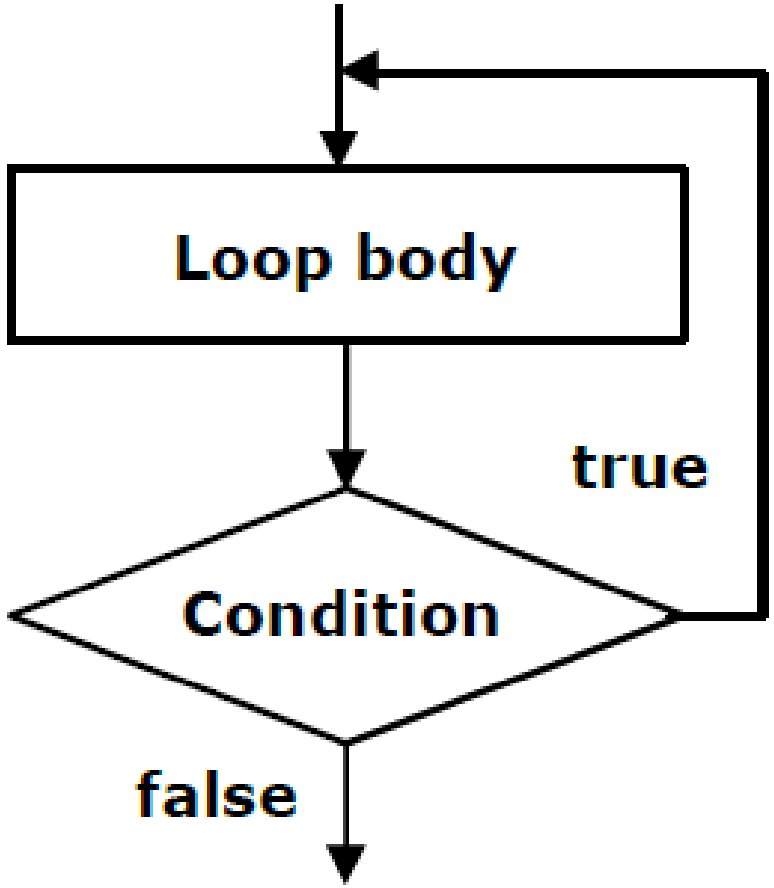
Loop (while loop)

.

```
int x = 0;
while (x < 5)
{
      Console.Write(x + " ");
}
```

```
0 1 2 3 4
```

Loop (do-while loop)



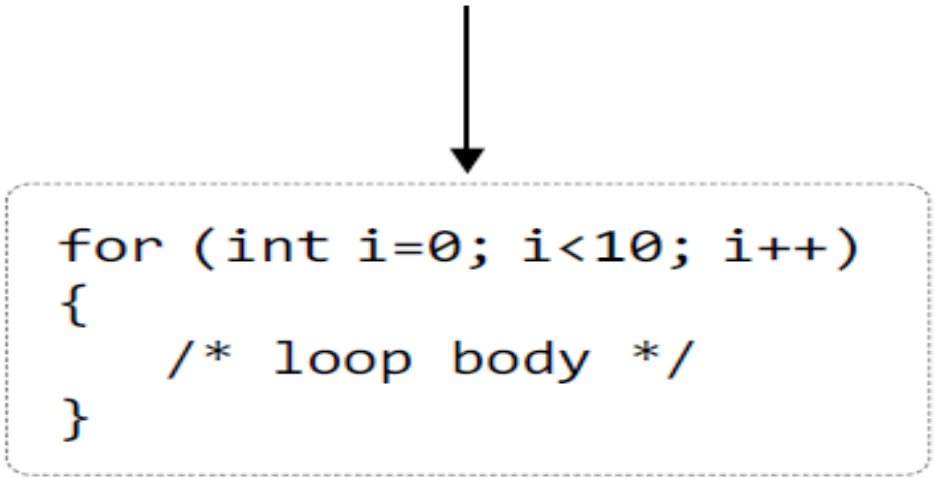
Loop (do-while loop)

```
int y = 0;
do
{
    Console.Write(y + " ");
}
while (y < 5);
```

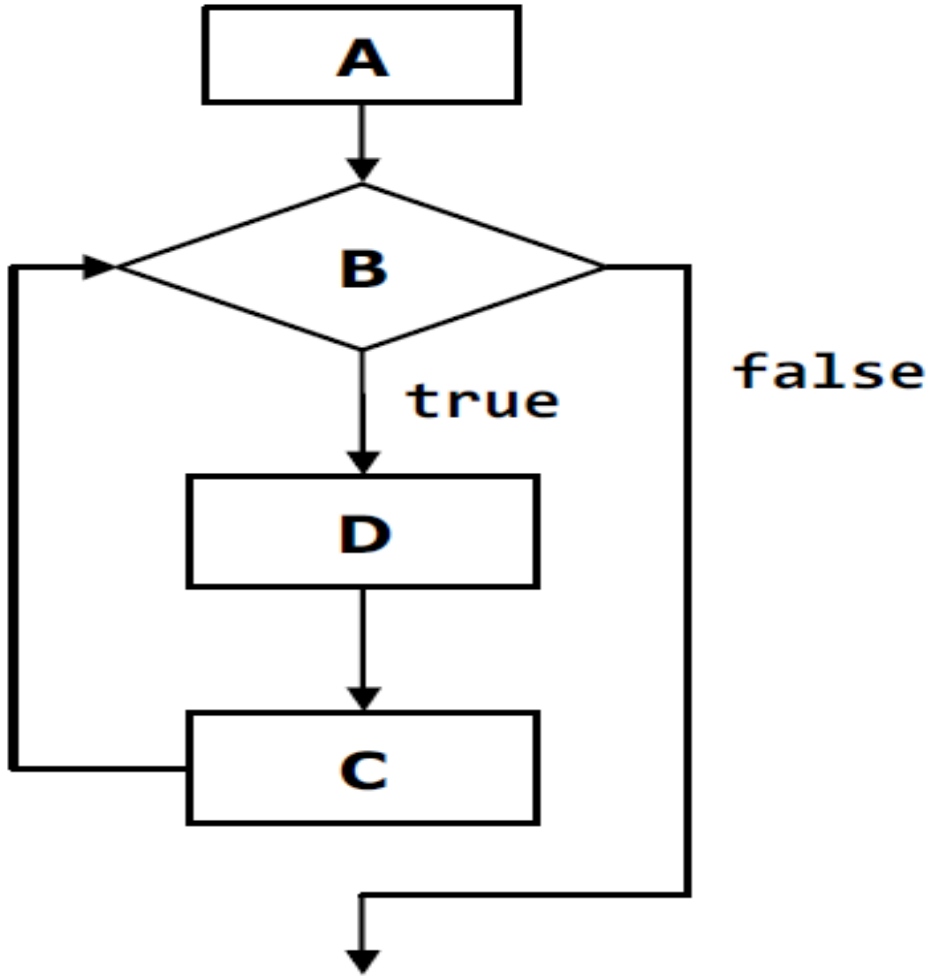
```
0 1 2 3 4
```

Loop (for loop)

```
for (A; B; C)
{
    D;
}
```



```
for (int i=0; i<10; i++)
{
    /* loop body */
}
```



Loop (for loop)

```
for (int x = 0; x < 5; x++ )
{
     Console.Write(x + " ");
}
```

```
0 1 2 3 4
```