Principle of Programing

3. Algorithm, operation and problem solving

Lecturers:

Sipan M. Hameed

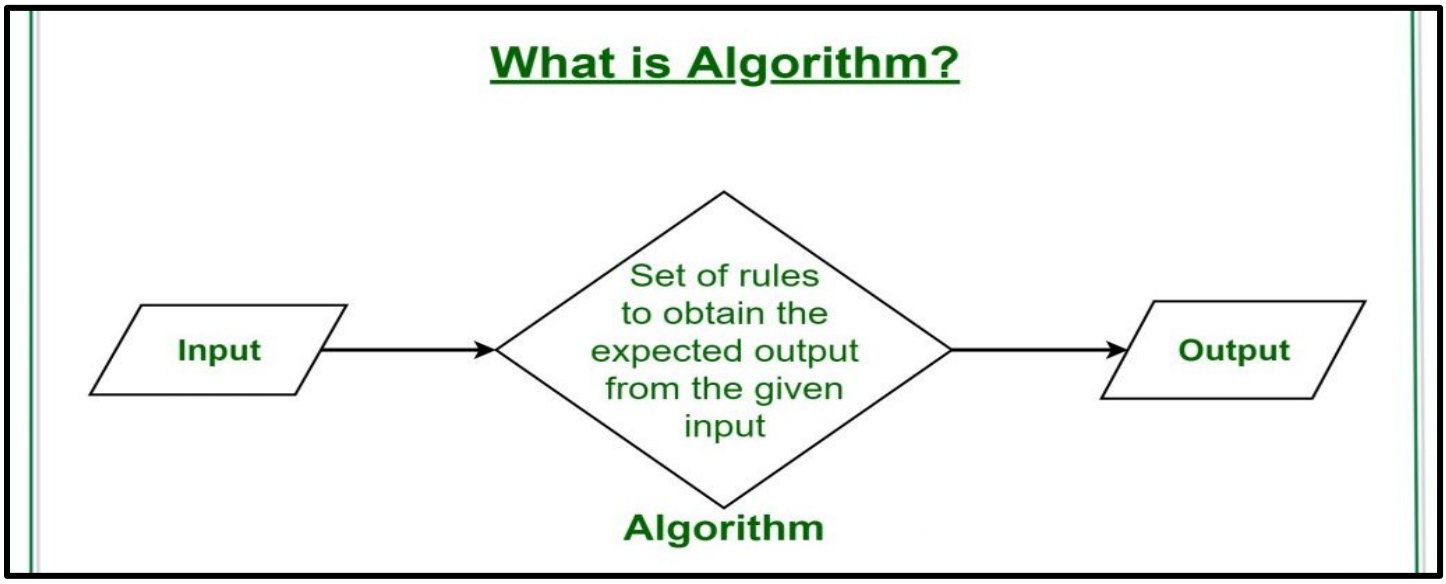
www.sipan.dev

2024 - 2025

1. Table of Contents

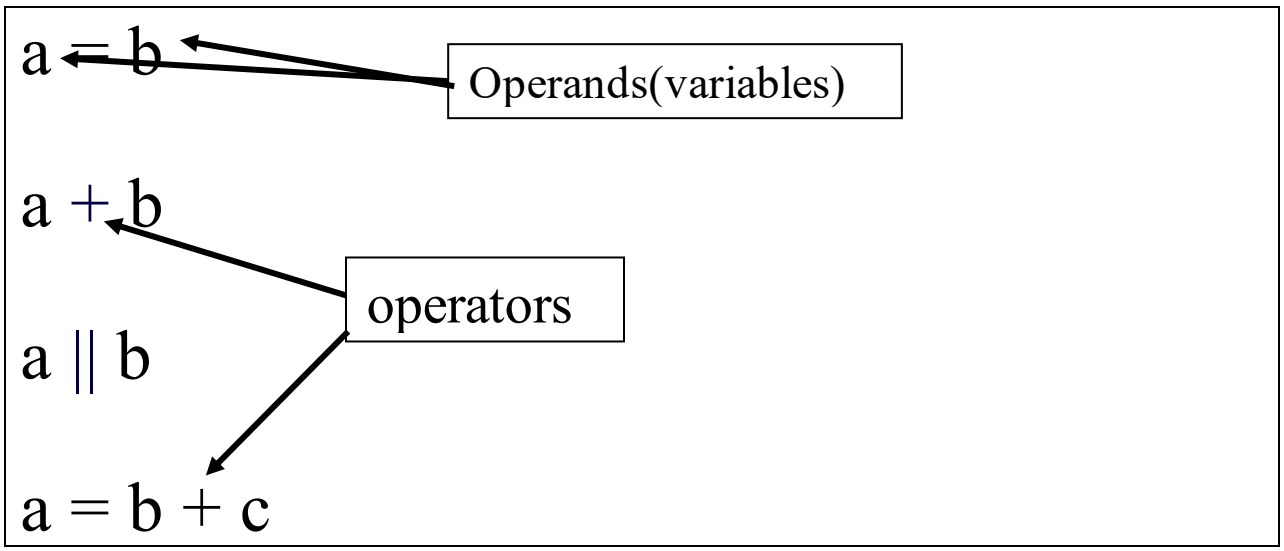
3. Algorithm

The word Algorithm means a process or set of rules to be followed in calculations or other problem-solving operations". Therefore, Algorithm refers to a set of rules/instructions that step-by-step define how a work is to be executed upon in order to get the expected results.



3.1 Expressions

An expression is a sequence of operators and their operands, that specifies a computation. Expression evaluation may produce a result (e.g., evaluation of 2+2 produces the result 4) and may generate side-effects (e.g. evaluation of (`Console.WriteLine("Hello World!");`) prints the `"Hello World!"` on the standard output.Examples:



in the above example the **operators** are**: (=, +, ||)**
the **operands(variable)** are:**(a, b, c)**

3.2 C# Operators

Operators are used to perform operations on variables and values. In the example below, we use the + operator to add together two values:

Example

```
int x = 100 + 50;
```

Although the + operator is often used to add together two values, like in the example above, it can also be used to add together a variable and a value, or a variable and another variable:

Example

```
int sum1 = 100 + 50;        // 150 (100 + 50)
int sum2 = sum1 + 250;      // 400 (150 + 250)
int sum3 = sum2 + sum2;     // 800 (400 + 400)
```

3.2.1 Arithmetic Operators

Arithmetic operators are used to perform common mathematical operations:

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	x++
--	Decrement	Decreases the value of a variable by 1	x--

3.2.2 C# Assignment Operators

Assignment operators are used to assign values to variables.

In the example below, we use the **assignment** operator (=) to assign the value **10** to a variable called **x**:

Example

```
int x = 10;
```

The **addition assignment** operator (+=) adds a value to a variable:

Example

```
int x = 10;
x += 5;
```

A list of all assignment operators:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

3.2.3 C# Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

3.2.4 C# Logical Operators

Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

3.3 use parentheses ( ) by default

We should note that if there is any possibility of confusion in any expression that involves multiple operators, you should use parentheses whether they are needed or not. They don't do any harm, and they guarantee the expression does what you want, even if you've made a mistake with precedence. Also, they make it clear to anyone reading the listing what you intended. For example:

Math equation:

$$a = \frac{x^2}{7} + y^3 - \frac{18}{2m}$$

C# code:

```
a = ( ( x*x ) / 7.0 )  +  ( y*y*y ) - ( 18.0 / (2.0*m) );
```

3.4 Practical examples

3.4.1 Example: Mathematical Operators

All arithmetic operators compute the result of specific arithmetic operation and returns its result. The arguments are not modified.

Note:

- if you have complex math equation define all variables and Literals as (**double**).
- Define Operators and use **parentheses ( )** by default to determine Operator Precedence.

Math formula: $$a = \frac{x^2}{7} + y^3 - \frac{18}{2m}$$

C# syntax:

```
a = ( ( x*x ) / 7.0 )  +  ( y*y*y ) - ( 18.0 / (2.0*m) );
```

```
using System;
namespace ex21
{
    class Program
    {
        static void Main()
        {
            double a, x, y, m;
            x = Convert.ToDouble(Console.ReadLine());
            y = Convert.ToDouble(Console.ReadLine());
            m = Convert.ToDouble(Console.ReadLine());
            a = ((x * x) / 7) + y * y * y - (18 / (2 * m));
            Console.WriteLine("a = {0}",a);

        }
    }
}
```

- Write programs to compute the following equations using basic **input/output**

$$p = \frac{5+2y^2}{x^3} \qquad f = \frac{4}{5}3xy^3 \qquad h = 8a^3 + \frac{b^2c^3}{m^4}$$

$$f = x^2 + 3y^3 - \frac{r^4 - 3.89}{3g^2}$$

3.4.2 Example: Comparison operations-1

Note:

- the result of Comparison operation is always **bool (true of false).**

```
using System;
namespace ex22
{
    class Program
    {
        static void Main()
        {
            int a, b;
            bool z;

            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());

            z = a == b;
            Console.WriteLine("a == b = {0}",z);
            z = a >= b;
            Console.WriteLine("a >= b = {0}", z);
            z = a <= b;
            Console.WriteLine("a <= b = {0}", z);
            z = a != b;
            Console.WriteLine("a != b = {0}", z);
            z = a < b;
            Console.WriteLine("a < b = {0}", z);
            z = a > b;
            Console.WriteLine("a > b = {0}", z);
        }
    }
}
```

Output:

```
2
7
a == b = False
a >= b = False
a <= b = True
a != b = True
a < b = True
a > b = False
```

3.4.3 Example: Logical operations-1

Enter an integer number and check the following cases:

- is positive number
- is positive number
- between 0 and 100
- is not between 0 and 100

sol:

```
using System;
namespace ex22
{
    class Program
    {
        static void Main()
        {
            int a;
            bool z;

            a = Convert.ToInt32(Console.ReadLine());

            z = a >= 0;
            Console.WriteLine("a is positive number:{0} ", z);
            z = a < 0;
            Console.WriteLine("a is positive number :{0}", z);
            z = a >= 0 && a <=100;
            Console.WriteLine("a between 0 and 100: {0}", z);
            z  = !(a >= 0 && a <= 100);
            Console.WriteLine("a is not between 0 and 100: {0}", z);

        }
    }
}
```

Output:

```
22
a is positive number:True
a is positive number :False
a between 0 and 100: True
a is not between 0 and 100: False
```

3.4.4 Example: Comparison operations-2

```
using System;
namespace ex22
{
    class Program
    {
        static void Main()
        {
            int a, b;
            bool z;

            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());

            z = a == b;
            Console.WriteLine("a == b = {0}",z);
            z = a >= b;
            Console.WriteLine("a >= b = {0}", z);
            z = a <= b;
            Console.WriteLine("a <= b = {0}", z);
            z = a != b;
            Console.WriteLine("a != b = {0}", z);
            z = a < b;
            Console.WriteLine("a < b = {0}", z);
            z = a > b;
            Console.WriteLine("a > b = {0}", z);
        }
    }
}
```

Output:

```
3
8
a == b = False
a >= b = False
a <= b = True
a != b = True
a < b = True
a > b = False
```

3.4.5 Example: Assignment and increment/decrement Operators

```
using System;
namespace ex22
{
    class Program
    {
        static void Main()
        {
            int a, b;
            bool z;

            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("a = {0}",a);
            a++;

            Console.WriteLine("a++   = {0}",a);
            Console.WriteLine("b = {0}", b);
            b = b + 10;
            Console.WriteLine("b = b + 10  = {0}",b);
            b += 10;
            Console.WriteLine("b += 10  =  {0}", b);


        }
    }
}
```

Output:

```
3
8
a = 3
a++   = 4
b = 8
b = b + 10  = 18
b += 10  =  28
```