



# Computer I

---



## **Lecture No. 3**

### **Algorithms Development, Pseudo-code and Flowchart , Arithmetic and Logical Operators**

Al-Mustaqbal University College of Engineering & Technology  
Biomedical Engineering Department  
MSc. in Computer Engineering: Hamza Waleed Hamza

# What is an algorithm?

- ❖ In computer programming, an algorithm is a set of well-defined instructions to solve a particular problem.
- ❖ It takes a set of input and produces a desired output.

Step 1: Start

1. بداية

Step 2: Define the variables

2. تعريف المتغيرات

Step 3: Read values of variables

3. قراءة قيمة كل متغير

Step 4: Process

4. معالجة

- math operations
- logic operations
- comparisons

▪ عمليات رياضية

▪ عمليات منطقية

▪ مقارنات

Step 5: Display result

5. عرض/طباعة الناتج

Step 6: End

6. نهاية

# Algorithm to add two numbers







- ❖ We need to first write the steps (sequence of actions) that lead to performing the task.
- ❖ For example, an algorithm to add two numbers:
  - Take two number inputs
  - Add numbers using the + operator
  - Display the result

Step #	Description
Step1	Start
Step2	Declare or define variables <code>num1, num2, sum;</code>
Step3	Read values num1 and num2
Step4	Add num1 and num2 and assign the results to sum <code>sum ← num1 + num2;</code>
Step5	Print sum
Step6	End

# Flowchart Symbols

- A Flowchart is a graphical representation that shows the behavior (workflow) of an algorithm.

Flowcharts use standard shapes including the following:

Symbol	Name	Description
	Start / End	An oval represents a <b>start</b> or <b>end</b> point
	Arrows	A line is a connector that shows flow direction between the representative shapes
	Input /Output	A parallelogram represents <b>input</b> or <b>output</b>
	Process	A rectangular represents a <b>process</b> (calculation)
	Decision	A diamond indicates a <b>decision</b> (comparison)
	Connector	A circle is used to combine one part of the flowchart with another part

# Adding two numbers

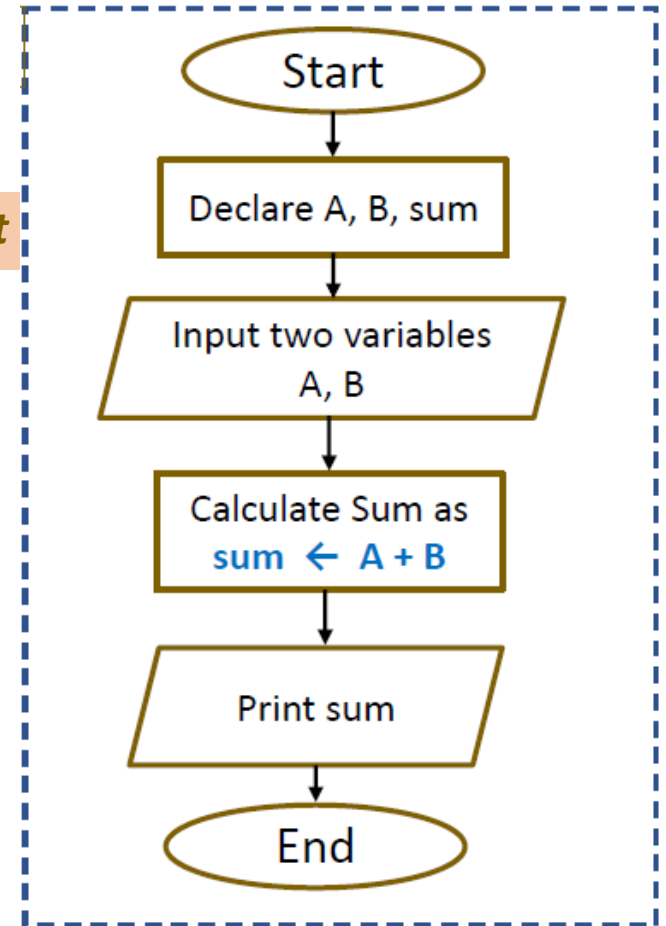
- ✓ Write the pseudo-code for algorithm to sum two numbers. Then draw the equivalent flowchart.
- Flowcharts use standard shapes including the following:

**Ans:**

*Pseudo-code Algorithm*

Step 1: START  
Step 2: DECLARE variables num1, num2, sum;  
Step 3: READ variables num1, num2;  
Step 4: CALCULATE sum  
           $\text{sum} = \text{num1} + \text{num2};$   
Step 5: PRINT sum  
Step 6: END

*Flowchart*



**Note:**

- The **pseudo-code** and **flowchart** of an algorithm is not written for a particular programming language.
- They can be used to plan a solution before coding it.

# Difference between pseudo-code and flowchart

## Main differences between Pseudo-code and Flowchart

- ❖ The pseudo-code is a high-level description of an algorithm while the flowchart is a graphical representation of an algorithm.
- ❖ An algorithm is a set of instructions for solving a problem or accomplishing a task.
- ❖ Every computerized device uses algorithms, which cut the time required to do things manually.
- ❖ Calculate the average of three input numbers.

### Pseudo-code Algorithm

Step 1: Start

Step 2: Declare variables Ave;

Step 3: Read variables num1, num2, num3;

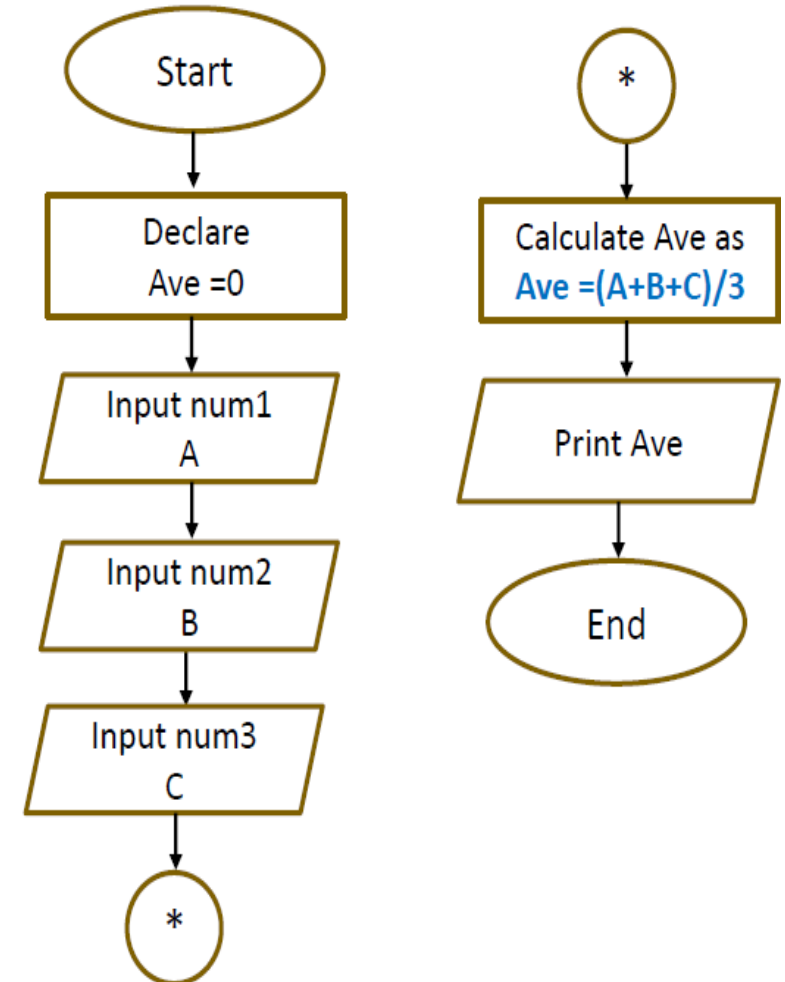
Step 4: Calculate Ave

$Ave = (num1 + num2 + num3)/3;$

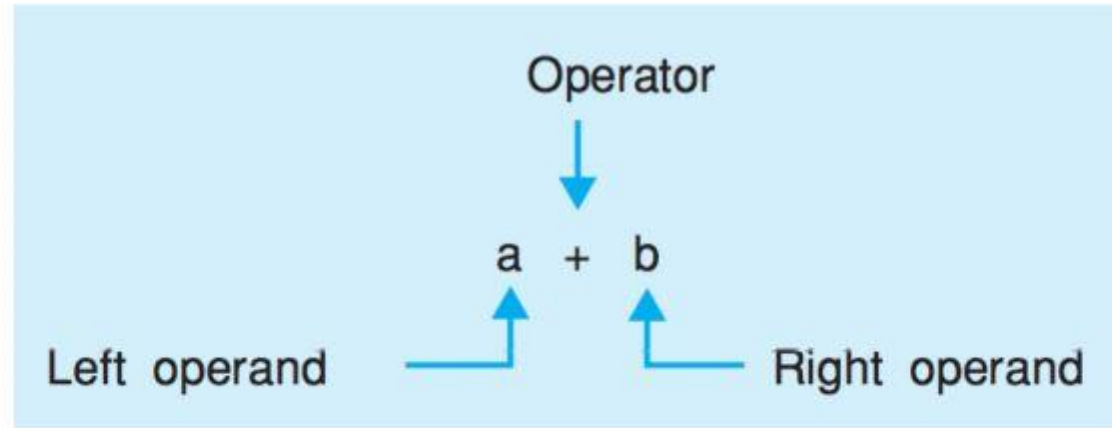
Step 5: Print Ave

Step 6: End

### Flowchart



# Binary Arithmetic Operators



Operator	Meaning
+	addition
-	subtraction
*	multiplication
/	division
%	remainder
=	assignment

# Binary Arithmetic Operators

- ❑ Arithmetic operators are used to perform calculations. You should be aware of the following:

- Divisions performed with integral operands will produce integral results;

**for example,  $7/2 = 3$**

- If at least one of the operands is a floating-point number, the result will also be a floating-point number;

**for example,  $7.0/2 = 3.5$**

- Remainder division is only applicable to integers and returns the remainder of an integral division;

**for example,  $7\%2 = 1$**

- Assignment operator  $=$ , assigns the value of a variable to an expression. In expressions of this type the variable must be placed on the left and the assigned value on the right of the assignment operator;

**for example,  $z = 7.5$  ,  $i = j = 9$**



# Example

```
#include <iostream>
using namespace std;
int main()
{
double x, y;
cout << "\nEnter two floating-point values: ";
cin >> x >> y;
cout << "The average of the two numbers is: "
<< (x + y)/2.0 << endl;
return 0;
}
```

# Expressions

**An expression consists of only:**

- ☐ one constant.
- ☐ one variable.
- ☐ one function call .
- ☐ a combination of operators and operands.
- ☐ Expressions return values.

Examples:

Int a(4) ; double x(7.9);

a \* 512           // Type int

1.0 + sin(x)     // Type double

x - 3            // Type double, since one // operand is of type double

2 + 7 \* 3        // Adds 2 and 21

(2 + 7) \* 3      // Multiplies 9 by 3

**Note :** Normal mathematical rules (multiplication before addition) apply when evaluating an expression, i.e. the \*, /, % operators have higher precedence than + and -.

# Unary Arithmetic Operators

There are four unary arithmetic operators: the sign operators + and -, the increment operator ++, and the decrement operator --.

**Example:** `int n = -5; cout << -n; // Output: 5`

Operator	Meaning
+	Plus sign
-	Minus sign
++	increment Operator
--	decrement Operator

# Increment / Decrement Operators

- ❑ The increment operator `++` modifies the operand by adding 1 to its value.
- ❑ The decrement operator `--` modifies the operand by reducing the value of the operand by 1.
- ❑ Given that `i` is a variable, both `i++` (postfix notation) and `++i` (prefix notation) raise the value of `i` by 1. In both cases the operation `i = i + 1` is performed.
- ❑ **However:**
  - ✓ `++i` is incremented first and the new value of `i` is then applied,
  - ✓ `i++` the original value of `i` is applied before `i` is incremented.

# Prefix and Postfix Notation

```
#include <iostream>
using namespace std;
int main()
{
    int i=2, j=8;
    cout << i++ << endl;    // Output: 2
    cout << i << endl;      // Output: 3
    cout << j-- << endl;    // Output: 8
    cout << j << endl;      // Output: 7
    cout << --j << endl;    // Output: 6
    cout << j << endl;      // Output: 6
    return 0;
}
```

# Relational Operators

- ❑ Each comparison in C++ is a bool type expression with a value of true or false, where true means that the comparison is correct and false means that the comparison is incorrect.

Operator	Meaning
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
==	equal
!=	unequal

## Examples

Comparison	Result
5 >= 6	false
1.7 < 1.8	true
4 + 2 == 5	false
2 * 4 != 7	true

# Logical Operators

- ❑ The logical operators comprise the Boolean operators:
  - ✓ && (AND)
  - ✓ || (OR)
  - ✓ ! (NOT)
- ❑ They can be used to create compound conditions and perform conditional execution of a program depending on multiple conditions.
- ❑ A logical expression results in a value false or true, depending on whether the logical expression is correct or incorrect, just like a relational expression.

A	B	A && B	A    B
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	false

A	!A
true	false
false	true

# Examples of Logical Operators

**Note:** A numeric value, such as  $x$  or  $x+1$ , is interpreted as “false” if its value is 0. Any value other than 0 is interpreted as “true”.

x	y	Logical Expression	Result
1	-1	$x \leq y \    \ y \geq 0$	false
0	0	$x > -2 \ \&\& \ y == 0$	true
-1	0	$x \ \&\& \ !y$	true
0	1	$!(x+1) \    \ y - 1 > 0$	false



# Home work

**Homework 1:** What values do the following arithmetic expressions have?

a.  $3/10$

b.  $11\%4$

c.  $15/2.0$

d.  $3 * 7 \% 4$

e.  $7 \% 4 * 3$

**Homework 2:** The int variable x contains the number 7. Calculate the value of the following logical expressions:

a.  $x < 10 \ \&\& \ x \geq -1$

b.  $x++ == 8 \ || \ x == 7$

Thank You