
# Gujarat Power Engineering & Research Institute

TOPIC :- C Operators

PREPAREDBy-Shah Viraj

# C Operators and Some Differences

**Contents**

**Operator:**

- Arithmetic Operators
- Increment & Decrement Operators
- Special Operators
- Conditional Operators

**Difference:**

- Data and Information
- Compiler and Interpreter
- System software and  Application Software
- Algorithm and Flowchart
- Array and Structure
- Call by Value and Call by Reference

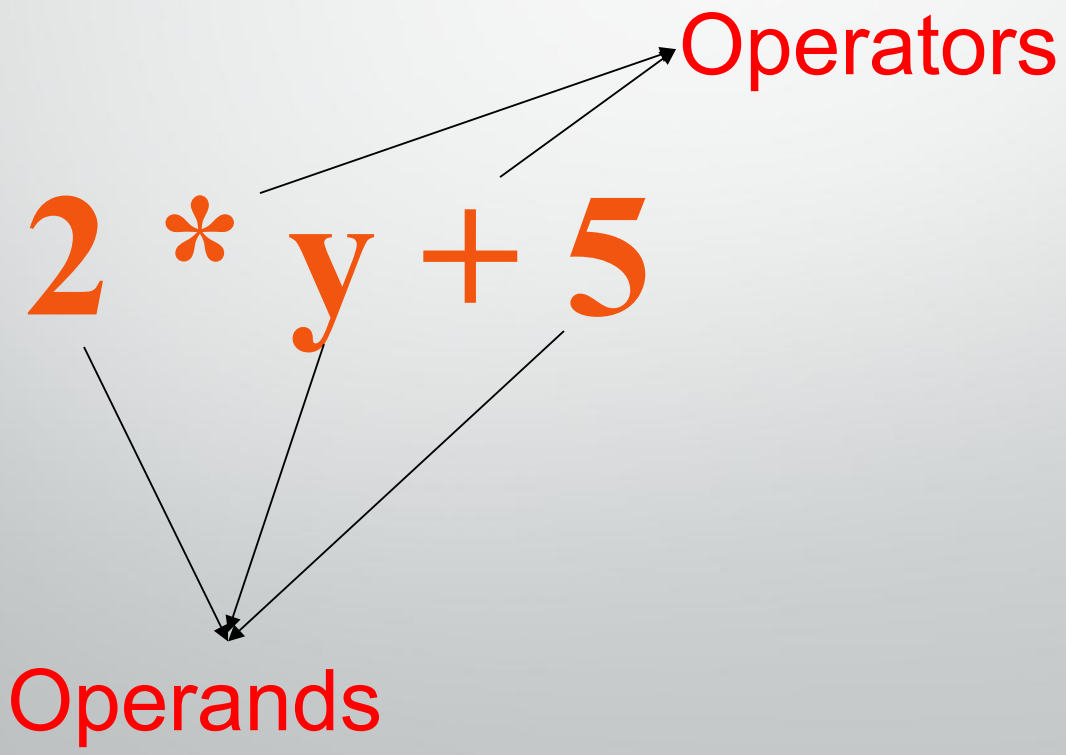
# OPERATORS

- **Operators** are symbols which take one or more **operands** or **expressions** and perform arithmetic or logical computations.
- **Operands** are **variables** or **expressions** which are used in conjunction with operators to evaluate the expression.
- Combination of operands and operators form an **expression**.

# Expressions

**Combination of Operators and Operands**

Example



# TYPES OF OPERATORS

-C language provides following operators:-

Types

- Arithmetic Operators
- Relational Operators
- Assignment Operators

Types

- Conditional Operators
- Logical Operators
- Bitwise Operators

Types

- Increment and Decrement Operators
- Special Operators

# Arithmetic Operators

- As the name is given, arithmetic operators perform arithmetic operations.

- C language provides following arithmetic operators.

| Operator name | Meaning |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulus(Reminder after division) |

# ARITHMETIC OPERATORS

- These are "Binary Operators" as they take two operands.

- The operands must be "Numeric Value".

- When both the operands are integer , then the result is integer. But when one of them is floating point and other is integer then the result is floating point.

- Except "%" operators, all the arithmetic operators can be used with any type of numeric operands(either operands are integer or floating point), while "%" operator can only be used with integer data type only.

# SMALL EXAMPLE TO ILLUSTRATE THE USE OFARITHMETIC OPERATORS

```
#include<stdio.h>
#include<conio.h>
Void main()
{
        float a=4,b=2,sum,sub,mul,div;
        clrscr();
 sum=a+b;
 sub=a-b;
 mul=a*b;
 div=a/b;
 printf("summation of a and b=%f\nsubtractionof and b=
%f\nmultiplicationof and b=%f\ndivision of a and b=
%f,sum,sub,mul,div");
getch();
}
```

# Output of the following code will be like this:

```
summation of a and b=6.000000
subtraction of a and b=2.000000
multiplication of a and b=8.000000
division of a and b=2.000000
```

# CONDITIONAL OPERATORS

- C language provides "?" operator as a conditional operator.

- These are "Ternary Operators" as they take three operands.

- It is used as shown below:

**Syntax:**

Condition ? Expression1 : Expression2

-The statement above is equivalent to:

```
if (Condition)
    Expression1
else
    Expression2
```

# CONDITIONAL OPERATORS

- The operator "?" works as follows:

- Expression1 is evaluated first. If it is nonzero(true), then the expression2 is evaluated and becomes the value of the expression.

- If expression1 is false, expression3 is evaluated and its value becomes the value of the expression.

- It is to be noted that only one of the expression(either expression2 or expression3) is evaluated.

- Hence, there is no dought that ternary operators are the alternative use of "if..else" statement.

Example 1:

<u>if/else statement:</u>

```
if (total > 60)
        grade = 'P'
else
        grade = 'F';
```

<u>conditional statement:</u>

```
total > 60 ? grade = 'P': grade = 'F';
                        OR
grade = total > 60 ? 'P': 'F';
```

# Example 2:

## if/else statement:

```
if (total > 60)
        printf("Passed!!\n");
else
        printf("Failed!!\n");
```

## Conditional Statement:

```
printf("%s!!\n", total > 60? "Passed":"Failed");
```

# SPECIAL OPERATORS

- C supports some special operators such as comma operator, size of operator, pointer operators (& and *) and member selection operators( . and -> ).

**The comma operator:**

- The comma operator is used to link the related expressions together or in the other way, it is used to combine multiple statememts.

- A comma linked list of expressions are evaluated from **"left"** to **"right"**.

# SPECIAL OPERATORS

i.e.

expression_1, expression_2
expression_1 is evaluated first
expression_2 is evaluated second

For Example:

value=(x=10,y=5,x+y);

-First assigns the value 10 to x, then assigns 5 to y, and finally assigns 10(i.e. 10+5) to value.

- Since comma operator has the lowest precedence of all operators.

-The parentheses are necessary.

x = (y=3 , y+1);  x = 4

( ) needed

# SIZEOF OPERATOR

- The sizeof is a compile time operator.
- When used with an operand, it returns the number of bytes the operand occupies.
- The operand maybe a variable, a constant or a data type qualifier.

```
Int a,m;
m = sizeof(a);
Printf("size of a in bytes is %d",m);
```

**Output:**

```
size of a in bytes is 2
```

# INCREMENT AND DECREMENT OPERATORS

- Sometimes we need to increase or drcrease a variable by "one".

- We can do that by using assignment operator =. For example the statement a=a+1 or a+=1 increment the value of variable a by 1.

- In the same way the statement a=a-1 or a-=1 decrements the value of variable a by 1.

- But C language provides "Increment and Decrement Operators" for this type of operations.

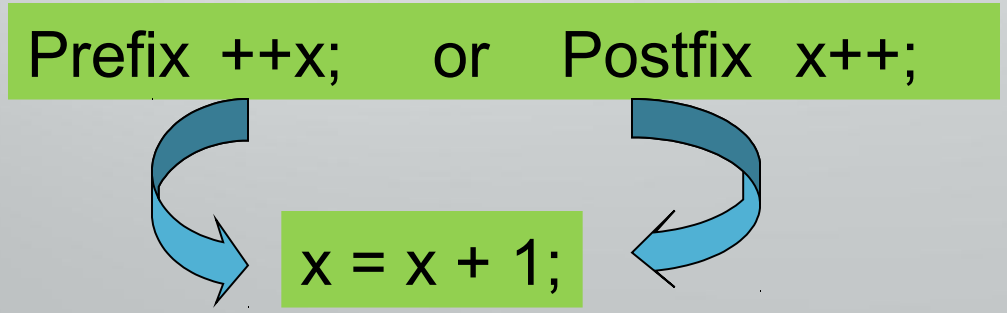
# INCREMENT AND DECREMENT OPERATORS

- These operators are "unary" as they take one operand only.

- The operand can be written before of after the operator.

- If a is a variable, then ++a is called "prefix increment", while a++ is called postfix increment.

- Similarly, --a is called "prefix decrement" and a– is called "postfix decrement".



If the ++ or – operator are used in an expression or assignment, then prefix notation and postfix notation give different values.

| | |
|---|---|
| a++ | Post-increment. After the result is obtained, the value of the operand is incremented by 1. |
| a++ | Post-decrement. After the result is obtained, the value of the operand is decremented by 1. |
| ++a | Pre-increment. The operand is incremented by 1 and its new value is the result of the expression. |
| --a | Pre-decrement. The operand is decremented by 1 and its new value is the result of the expression. |

# Simple example to understand the prefix and postfix increment:

```
int a=9,b;
b=a++;
printf("%d\n",a);
printf("%d",b);
```

The output would be

```
10
9
```

But, if we execute this code…

```
int a=9,b;
b=++a;
printf("%d\n",a);
printf("%d",b);
```

The output would be

```
10
10
```

# Simple example to understand the prefix and postfix decrement:

```
int a=9,b;
b=a--;
printf("%d\n",a);
printf("%d",b);
```

The output would be

```
8
9
```

But, if we execute this code…

```
int a=9,b;
b=--a;
printf("%d\n",a);
printf("%d",b);
```

The output would be

```
8
8
```

# DIFFERENCES

| Complier | Interpreter |
| --- | --- |
| Translates whole source code into object code. | Translate land execute line by line of source code. |
| User cannot see the result of program at translation time. | User can see the result of program at translation time. |
| It requires less main memory beacuase at the time of translation it place object code into secondary memory. Execution is not the job of compiler. This is the job of loader | It requires less main memory beacuase at the time of translation it place object code into main memory and execute it and produce the result. |

# ***DIFFERENCES***

| Complier | Interpreter |
| --- | --- |
| This process is faster than interpreter. | This process is slower than compiler. |
| Size of compiler program is smaller than interpreter beacuse it contain only translation procedure. | Size of interpreter program is larger than compiler beacuse it contain two tasks translation as well as loading. |

# DIFFERENCES

| Arrays | Structures |
| --- | --- |
| An array is a collection of related data elements of same type. | Structure can have elements of different types |
| Syntax: <data_type> array_name[size]; | **struct** struct_name<br>{<br>structure element 1;<br>structure element 2;<br>---------<br>---------<br>structure element n; } |
| An array is a derived data type | A structure is a programmer-defined data type |

# DIFFERENCES

| Arrays | Structures |
| --- | --- |
| Any array behaves like a built-in data types. All we have to do is to declare an array variable and use it. | But in the case of structure, first we have to design and declare a data structure before the variable of that type are declared and used. |
| Array elements are accessed by it's position or subscript. | Structure elements are accessed by its object as '.' operator. |
| Array element access takes less time in comparison with structures. | Structure element access takes more time in comparison with arrays. |

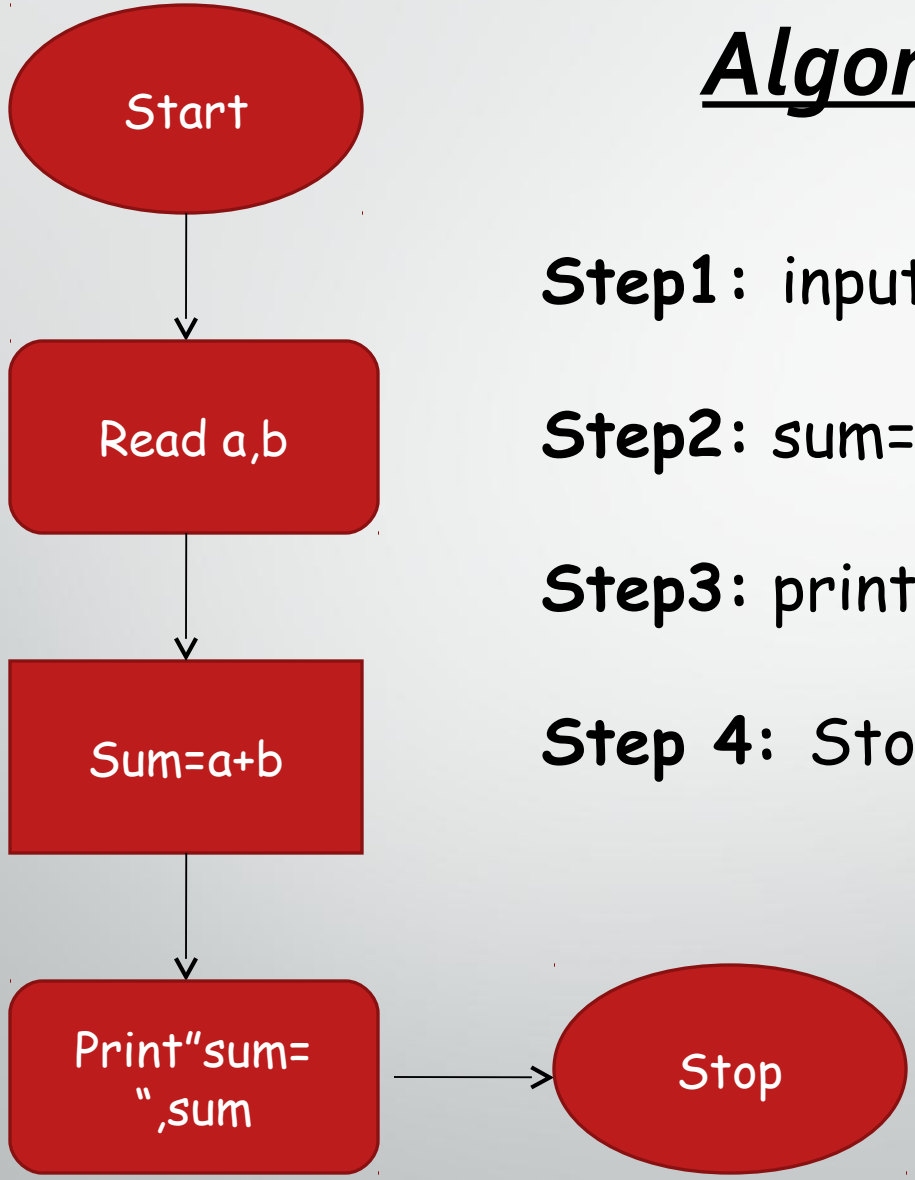
# DIFFERENCES

| Data | Information |
| --- | --- |
| Data is raw, unorganized facts that need to be processed. Data can be something simple and seemingly random and useless | When data is processed, organized, structured or presented in a given context so as to make it useful, it is called Information. |
| Each student's test score is one piece of data | The class' average score or the school's average score is the information that can be concluded from the given data. |
| **Data is always correct** (I can't be 29 years old and 62 years old at the same time) | But **information can be wrong** (there could be two files on me, one saying I was born in 1981, and one saying I was born in 1948). |

# DIFFERENCES

| Algorithm | Flow chart |
| --- | --- |
| An algorithm is a finite sequence of well defined steps for solving a problem in a systematic manner. | A flow chart is a graphical representation of sequence of any problem to be solved by computer programming |
| Difficult to show branching and looping. | Easy for branching and looping. |
| It does not includes any specific symbols. | Flowchart uses specific symbols for specific reasons. |

# Flow Chart:




# Algorithm:

**Step1:** input two numbers:a,b

**Step2:** sum=a+b

**Step3:** print"sum= ",sum

**Step 4:** Stop

# DIFFERENCES

| System Software | Application Software |
| --- | --- |
| System software is computer software designed to operate the computer hardware and to provide a platform for running application software. | Application software is computer software designed to help the user to perform specific tasks. |
| It is general-purpose software. | It is specific purpose software. |
| System Software Create his own environment to run itself and run other application. | Application Software performs in a environment which created by System/Operating System. |
| It executes all the time in computer. | It executes as and when required. |

# DIFFERENCES

| System Software | Application Software |
| --- | --- |
| System software is essential for a computer | System software is essential for a computer |
| The number of system software is less than application software | The number of application software is much more than system software |
| System software is written in low level language | Application software are usually written in high level language |
| Detail knowledge of hardware is required. | Detail knowledge of organization is required. |

| Call by value | Call by reference |
| --- | --- |
| call by value passes the copy of the variable to the function that process it | call by reference passes the memory reference of the variable (pointer) to the function |
| main ()<br>{<br>sum (10, 20);<br>//other code…<br>}<br>function sum (a, b)<br>{ //your code…..<br>} | main ()<br>{<br>sum (&a, &b);<br>//other code…<br>}<br>function sum (*a, *b)<br>{ //your code…..<br>} |
| call by value increases the memory assigned to the code because of the copy of variables at each reference | whereas call by reference utilizes the same reference of variable at each reference. |

# THANK YOU