What is an algorithm?

**An algorithm is an effective, efficient and best method which can be used to express solution of any problem within a finite amount of space and timeand in a well-defined formal language**. Starting from an initial state the instructions describe a process or computational process that, when executed, proceeds through a finitenumber of well-defined successive states, eventually producing "output"and terminating at a final ending state.

**In other words, we can say that**,

1. Step by step procedure for solving any problem is known as algorithm.
2. An algorithm is a finite set of instructions that, if followed, accomplishes a particular task.
3. An algorithm is a sequence of computational steps that transform the input into a valuable or required output.
4. Any special method of solving a certain kind of problem is known as algorithm.

**All Algorithms must satisfy the following criteria -**

1. **1) Input**  
   There are more quantities that are extremely supplied.
2. **2) Output**  
   At least one quantity is produced.
3. **3) Definiteness**  
   Each instruction of the algorithm should be clear and unambiguous.
4. **4) Finiteness**  
   The process should be terminated after a finite number of steps.
5. **5) Effectiveness**  
   Every instruction must be basic enough to be carried out theoretically or by using paper and pencil.

Properties of Algorithm

Simply writing the sequence of instructions as an algorithm is not sufficient to accomplish certain task. It is necessary to have following properties associated with an algorithm.

1. **Non Ambiguity**  
   Each step in an algorithm should be non-ambiguous. That means each instruction should be clear and precise. The instruction in any algorithm should not denote any conflicting meaning. This property also indicates the effectiveness of algorithm.
2. **Range of Input**  
   The range of input should be specified. This is because normally the algorithm is input driven and if the range of input is not being specified then algorithm can go in an infinite state.
3. **Multiplicity**  
   The same algorithm can be represented into several different ways. That means we can write in simple English the sequence of instruction or we can write it in form of pseudo code. Similarly, for solving the same problem we can write several different algorithms.
4. **Speed**  
   The algorithmis written using some specified ideas. Bus such algorithm should be efficient and should produce the output with fast speed.
5. **Finiteness**  
   The algorithm should be finite. That means after performing required operations it should be terminate.

**Advantages of algorithm**

1. It is a step-wise representation of a solution to a given problem, which makes it easy to understand.
2. An algorithm uses a definite procedure.
3. It is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge.
4. Every step in an algorithm has its own logical sequence so it is easy to debug.
5. By using algorithm, the problem is broken down into smaller pieces or steps hence, it is easier for programmer to convert it into an actual program

**Disadvantages of algorithm.**

1. Writing algorithm takes a long time.
2. An Algorithm is not a computer program, it is rather a concept of how a program should be.

## C++ Data Types

The data type specifies the size and type of information the variable will store:

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Size** | **Description** |
| int | 4 bytes | Stores whole numbers, without decimals |
| float | 4 bytes | Stores fractional numbers, containing one or more decimals. Sufficient for storing 7 decimal digits |
| double | 8 bytes | Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits |
| boolean | 1 byte | Stores true or false values |
| char | 1 byte | Stores a single character/letter/number, or ASCII values |

### **Operators In C++**

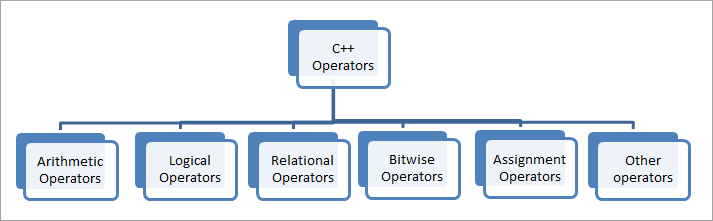
Operators form the basic foundation of any programming language. Without operators, we cannot modify or manipulate the entities of programming languages and thereby cannot produce the desired results. C++ is very rich in built-in operators which we will discuss in detail in this tutorial.

In C++ most of the operators are binary operators i.e. these operators require two operands to perform an operation. Few operators like ++ (increment) operator are the unary operator which means they operate on one operand only.

There is also a ternary operator in C++ called Conditional Operator which takes three operands. We will learn about this in detail in the later part of the tutorial.

### **Types Of Operators In C++**

**Operators in C++ are classified as shown below:**

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2019/04/c-operators.png)

#### **Arithmetic Operators**

Arithmetic operators are used for performing basic mathematical operations on operands.

**C++ supports the following arithmetic operations:**

| **Operator** | **Binary/unary** | **Description** |
| --- | --- | --- |
| + | Binary | Addition of two operands |
| - | Binary | Subtraction of two operands |
| \* | Binary | Multiplication of two operands |
| / | Binary | Division of two operands |
| % | Binary | Modulus operator – the result is the remainder of the division |
| ++ | Unary | Increment operator – increases the value of operand by 1 |
| -- | Unary | Decrement operator – decreases the value |

#### **Logical Operators**

Logical operators are used for evaluating a combination of conditions/constraints to get a resultant value. The result of the evaluation of a Boolean expression is Boolean which is either true or false.

**C++ supports the following logical operators:**

| **Operator** | **Description** |
| --- | --- |
| && | Logical AND: returns true if both conditions are true otherwise returns false. |
| || | Logical OR: returns true if one of the conditions is true. Returns false when both conditions are false. |
| ! | Logical NOT: negates the condition. |

#### **Logical Operators**

Logical operators are used for evaluating a combination of conditions/constraints to get a resultant value. The result of the evaluation of a Boolean expression is Boolean which is either true or false.

**C++ supports the following logical operators:**

| **Operator** | **Description** |
| --- | --- |
| && | Logical AND: returns true if both conditions are true otherwise returns false. |
| || | Logical OR: returns true if one of the conditions is true. Returns false when both conditions are false. |
| ! | Logical NOT: negates the condition. |

#### **Assignment Operators**

Assignment operator “=” is used to assigning a value to a variable. The LHS of the assignment operator is a variable and RHS is the value that is to be assigned to the variable. The value on the right side must be of the same type as that of the variable on the left-hand side.

Note the difference between ‘=’ and ‘==’ operators. The former is the assignment operator and the later is the equality operator.

**The below table gives us a description of these assignment operators.**

| **Operator** | **Description** |
| --- | --- |
| = | Assigns the value of RHS operand to LHS operand |
| += | Adds RHS operand to LHS operand and assigns the result in LHS operand. |
| -= | Subtracts RHS operand to LHS operand and assigns the result to LHS operand |
| \*= | multiplies RHS operand to LHS operand and assigns the result to LHS operand |
| /= | divides RHS operand to LHS operand and assigns the result to LHS operand |