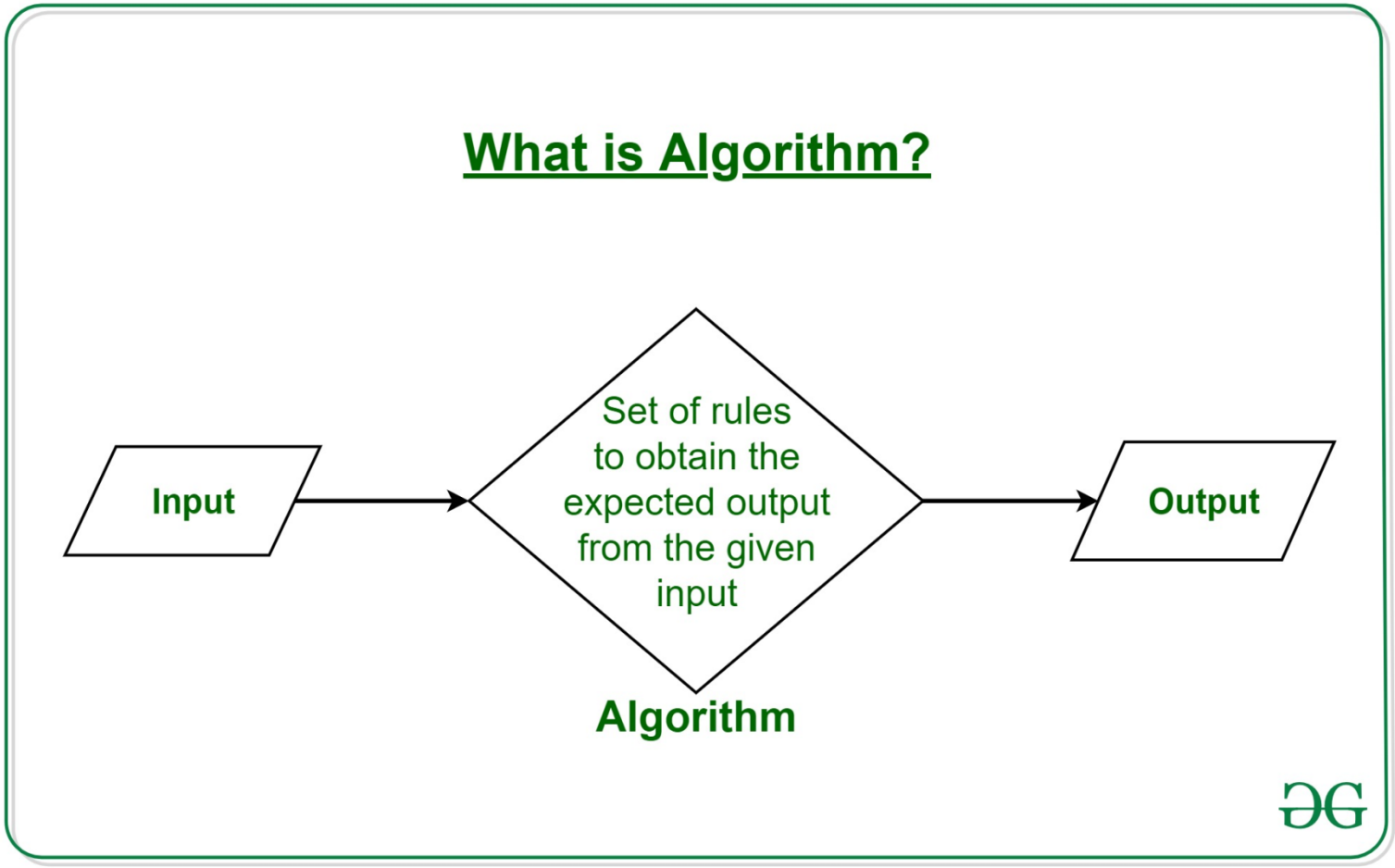
CONCEPT OF AN ALGORITHM

By:[Shreshtha Misra]
[CSE DEPT.]

What is an 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.
- Algorithms help to do a task in programming to get the expected output.
- The Algorithm designed are language-independent, i.e. they are just plain instructions that can be implemented in any language, and yet the output will be the same, as expected.

What is an algorithm?



Characteristics of an algorithm

- In order for some instructions to be an algorithm, it must have the following characteristics:
- **Clear and Unambiguous**: Algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
- **Well-Defined Inputs**: If an algorithm says to take inputs, it should be well-defined inputs.
- **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well.
- **Finite-ness:** The algorithm must be finite, i.e. it should not end up in an infinite loops or similar.

Characteristics of an algorithm

- **Feasible:** The algorithm must be simple, generic and practical, such that it can be executed upon with the available resources. It must not contain some future technology, or anything.
- **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be same, as expected.

How to Design an Algorithm?

- In order to write an algorithm, following things are needed as a pre-requisite:
- The **problem** that is to be solved by this algorithm.
- The **constraints** of the problem that must be considered while solving the problem.
- The **input** to be taken to solve the problem.
- The **output** to be expected when the problem the is solved.
- The **solution** to this problem, in the given constraints.
- Then the algorithm is written with the help of above parameters such that it solves the problem

Example of an algorithm

- **Example: Consider the example to add three numbers and print the sum.**
- **Step 1: Fulfilling the pre-requisites:** As discussed , in order to write an algorithm, its pre-requisites must be fulfilled.
  - **The problem that is to be solved by this algorithm**: Add 3 numbers and print their sum.
  - **The constraints of the problem that must be considered while solving the problem**: The numbers must contain only digits and no other characters.
  - **The input to be taken to solve the problem:** The three numbers to be added.

Example of an algorithm

- **The output to be expected when the problem is solved:** The sum of the three numbers taken as the input.
- **The solution to this problem, in the given constraints:** The solution consists of adding the 3 numbers. It can be done with the help of '+' operator, or bit-wise, or any other method.
- **Step 2: Designing the algorithm:** Now let's design the algorithm with the help of above pre-requisites:

Example of an algorithm

- **Algorithm to add 3 numbers and print their sum:**
- START
- Declare 3 integer variables num1, num2 and num3.
- Take the three numbers, to be added, as inputs in variables num1, num2, and num3 respectively.
- Declare an integer variable sum to store the resultant sum of the 3 numbers.
- Add the 3 numbers and store the result in the variable sum.
- Print the value of variable sum
- END
- **Step 3: Testing the algorithm by implementing it.** In order to test the algorithm, let's implement it in C language.

Algorithms to programs

- C program to add three numbers with the help of above designed algorithm

```
#include <stdio.h>
int main()
{
    int num1, num2, num3,sum;
    printf("Enter the 1st number: ");
    scanf("%d", &num1);
    printf("%d\n", num1);

    printf("Enter the 2nd number: ");
    scanf("%d", &num2);
    printf("%d\n", num2);
```

Algorithms to programs

```
 printf("Enter the 3rd number: ");
 scanf("%d", &num3);
 printf("%d\n", num3);

 sum = num1 + num2 + num3;
 printf("\nSum of the 3 numbers is: %d", sum);
  return 0;
}
```

How to Analyze an Algorithm?

- For a standard algorithm to be good, it must be efficient. Hence the efficiency of an algorithm must be checked and maintained. It can be in two stages:

  - **Priori Analysis:** "Priori" means "before". Hence Priori analysis means checking the algorithm before its implementation. In this, the algorithm is checked when it is written in the form of theoretical steps. This Efficiency of an algorithm is measured by assuming that all other factors, for example, processor speed, are constant and have no effect on the implementation. This is done usually by the algorithm designer. It is in this method, that the Algorithm Complexity is determined.

How to Analyze an Algorithm?

- **Posterior Analysis:** “Posterior” means “after”. Hence Posterior analysis means checking the algorithm after its implementation. In this, the algorithm is checked by implementing it in any programming language and executing it. This analysis helps to get the actual and real analysis report about correctness, space required, time consumed etc.

What is Algorithm Complexity?

- An algorithm is defined as complex based on the amount of Space and Time it consumes. Hence the Complexity of an algorithm refers to the measure of the Time that it will need to execute and get the expected output, and the Space it will need to store all the data (input, temporary data and output). Hence these two factors define the efficiency of an algorithm.
- **The two factors of Algorithm Complexity are:**
  - **Time Factor**: Time is measured by counting the number of key operations such as comparisons in the sorting algorithm.
  - **Space Factor**: Space is measured by counting the maximum memory space required by the algorithm.

Thank You