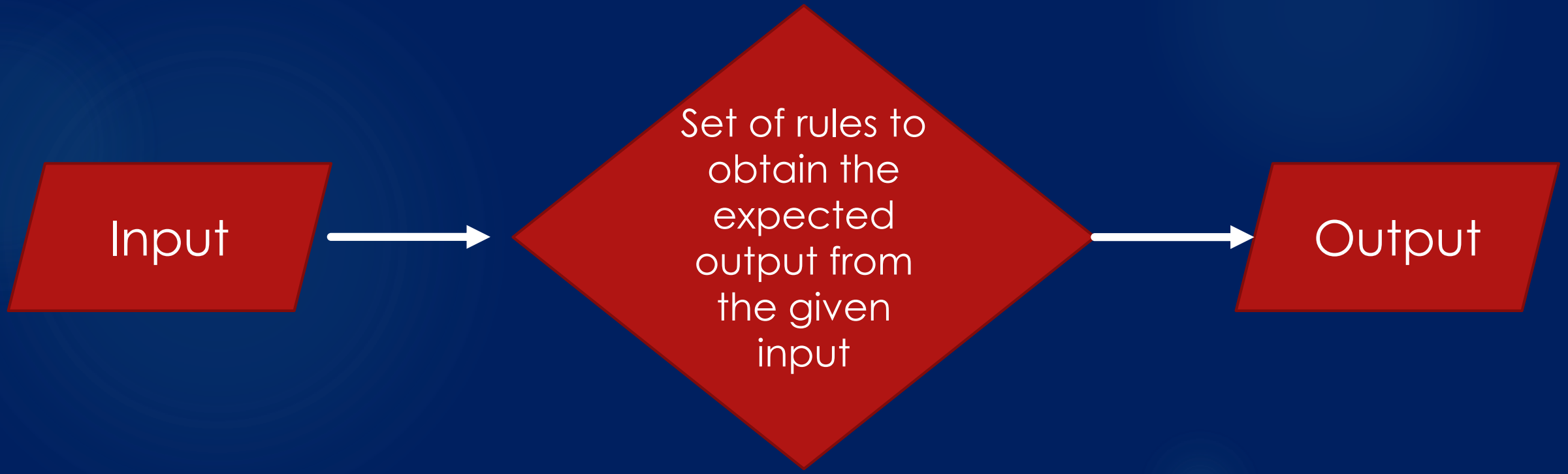
Introduction to algorithms

Algorithm

- Algorithm: An algorithm is a set of well-defined instructions to solve a particular problem. It takes a set of input(s) and produces the desired output.



Algorithm (cont..)

- Algorithms generally have the following characteristics.
  1. **Finiteness**: Algorithm must complete after a finite number of instruction have been executed.
  2. **Input**: The algorithm received input, Zero or more quantities are externally supplied
  3. **Output**: the algorithm produces output, at least one quantity is produced
  4. **Precision**: The steps are precisely stated. Each instruction is clear and unambiguous.
  5. **Feasibility**: it must be feasible to execute each instruction
  6. **Flexibility**: it should also be possible to make changes in the algorithm without putting so much effort on it.
  7. **Generality**: The algorithm applies to a set of inputs.

Types of Algorithms (few important)

- Some important types of algorithms are discussed here:
- 1. Brute Force Algorithm:
  - It is the simplest approach to problem-solving
  - Brute Force Algorithms are exactly what they sound like Straight Forward methods of solving a problem that rely on absolute computing power and trying every possibility rather than advanced techniques to improve efficiency.
- 2. Recursive Algorithm:
  - A recursive algorithm is based on recursion (call the same task again and again)
  - In this case, a problem is broken into several sub-parts and called the same function again and again

Types of Algorithms (few important)

- 3. Divide and Conquer Algorithm:
  - This algorithm breaks a problem into sub-problems, solves a single sub-problem and merges the solutions together to get the final solution.
  - It consists of the following three steps:
    1. Divide
    2. Solve
    3. Combine
- 4. Searching Algorithms:
  - Searching algorithms are the ones that are used for searching elements or groups of elements from a particular data structure (e.g. Linear Search, Binary Search).
  - These algorithms could be of different types based on their approach or the data structure in which the element should be found.

Types of Algorithms (few important)

- 5. Sorting Algorithm:
  - Sorting is arranging a group of data in a particular order (ascending/descending) according to the requirement.
  - The algorithms which help in performing this function are called sorting algorithms.
  - Generally sorting algorithms are used to sort groups of data in an increasing (ascending) or decreasing (descending) manner.

How to design an Algorithm

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

How to design an Algorithm (Example)

- Write a program to calculate the Factorial of the given number by using for loop and if condition.
- **Problem**: Calculate the Factorial e.g. 5! = 5x4x3x2x1 = 120
- **Constraint**: use for loop and if condition
- **Input**: any given number by the user
- **Output**: factorial
- **Solution**:

  Lets try on IDE

Basic Data Types and Variables

Data types

- in programming languages define the type of data that a variable can hold.
- These types determine what kind of operations can be performed on the data and
- how much memory will be allocated for it.
- Here are the basic data types commonly found in most programming languages (like C++, Java, Python, etc.):

1. Integer (int):

   Used to store whole numbers (both positive and negative).

   e.g. int age = 25;

   range: Typically from -2,147,483,648 to 2,147,483,647 (varies by system).

Data types (Cont.…)

2. Floating-point (float, double):

   Used to store real numbers with decimal points.

   float: Less precision (usually up to 7 decimal places).

   e.g. float height = 5.9;

   double: More precision (up to 15 decimal places).

   e.g. double weight = 72.56789;

3. Character (char):

   Used to store a single character (letters, digits, symbols).

   e.g. char grade = 'A';

   Typically 1 byte, stores the ASCII value of the character.

Data types (Cont.…)

3. Boolean (bool):

   Used to store truth values true or false.

   e.g. bool isStudent = true;

4. String:

   Used to store sequences of characters (a collection of char).

   e.g. string name = “John";

Variables

- A **variable** is a container that stores data. You can assign a value to a variable and later use or modify it.

Declaring a Variable

syntax: <data_type> variable_name = value;

my suggestion:

- Use names that explain the purpose of the variable;

e.g. Instead of declaring the variable as **int x** (which holds the age), we can declare it as **int age**.

Variables

- Follow Naming Conventions

  in most of the programming languages, camelCase is common for variable names (start with a lowercase letter, and subsequent words are capitalized)

  e.g.

  ```
  int totalMarks;
  float averageHeight;
  bool isLoggedIn;
  ```

- Avoid Single Letter Names

  Except for variables with very small scopes (like in loops), avoid single-letter names like a, b, or x, as they don't convey meaning

  e.g. int p; instead you may use int producPrice;

Variables

- Avoid Abbreviations

  Abbreviations can be confusing. Instead, use complete words for better clarity.

  e.g. instead of int numStd;

  use int numberOfStudents;

- Avoid Keywords as Variable Name

  Don't use keywords or reserved words

  e.g. instead of int return;

  use int returnValue;

- Use Boolean Names that Suggest True/False Meaning

  When defining boolean variables, choose names that clearly indicate a true or false condition.

  e.g. bool isComplete;

  bool hasError;

  bool isStudent;

Variables

- Use Boolean Names that Suggest True/False Meaning

  When defining boolean variables, choose names that clearly indicate a true or false condition.

  e.g. bool isComplete;

  bool hasError;

  bool isStudent;