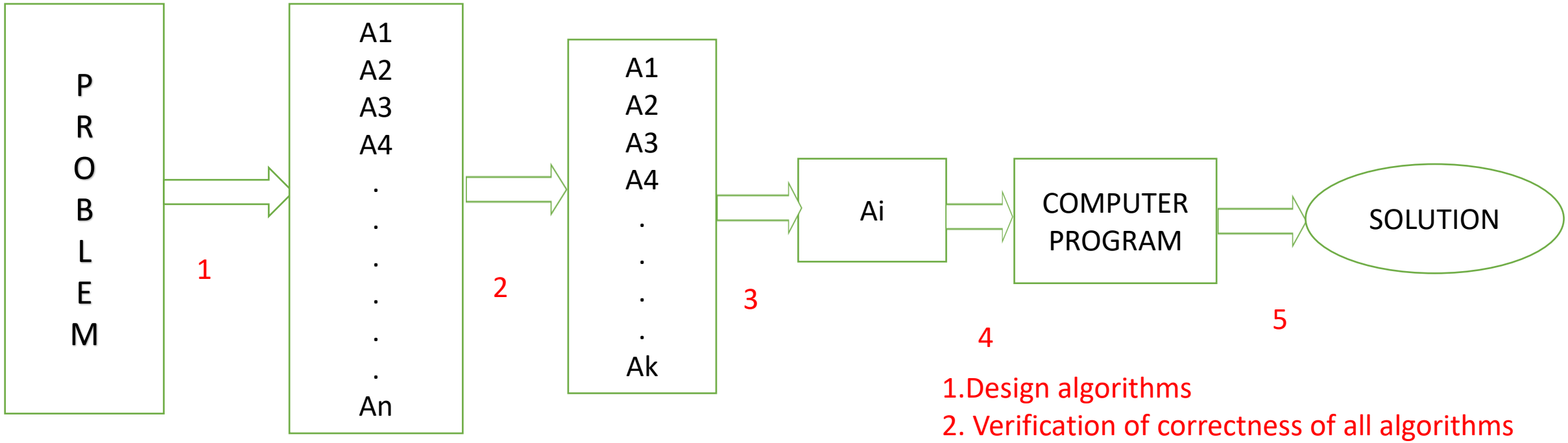
Introduction to Algorithm Design

Algorithms are the threads that tie together most of the subfields of computer science.
by ***Donald Knuth***

Introduction to Algorithm Design



1. I/p
2. Condition/Constraints
3. O/P

1.Design algorithms
2. Verification of correctness of all algorithms
3. Performance Analysis of correct algorithms
4. Translate the most efficient algorithm into equivalent computer program
5. Execution of the program

Introduction to Algorithm Design

- What Is An Algorithm?
- Why study Algorithm?
- What's the Importance of Algorithms in Computer Programming?

Example:1

- You are thinking to study Engg
- But you are not sure about your rank
- What will you do
- You are defining a work plan according to your expectation (ITER,IIT,IIIT, NIT, CET etc.)

Example2

- You are planning to build a big house
- But at the same time, you are not sure whether the resources that you have are enough or not.
- What will you do?
- You will define a work plan that will ensure what ever resources available that will help to finish the building

Example3

You are thinking to attend your friend's marriage ceremony which is 100 km far from Bhubaneswar.

- but you have little time available (you have two days leave)
- It is quite obvious that you will get the shortest or fastest route that will get you to your destination.

Introduction to Algorithm Design

Important note ----------planning/work plan

Computer programming ----------Algorithm

Example:

--Instagram uses an algorithm which determines which posts to show you at the top of your feed, based on your browsing habits. This way, you spend less time looking for posts that you might be interested in.

-- **W**hen you search for something on Google, there are many results, even pages and pages of results. However, more often than not, you never have to venture off that first page of results, because the most important and relevant items are on that first page. This is because of a specific algorithm that Google uses in order to sort through and figure which websites are the most important, based on relevance and website rank.(Page Ranking Algorithm)

Introduction to Algorithm Design

What Is An Algorithm?

--- An **algorithm** is a set of instructions designed to perform a specific task.

--- **Algorithms** are simply a series of instructions that are followed, step by step, to do something useful or solve a problem.

--- An **algorithm** is a procedure or formula for solving a problem, based on conducting a sequence of specified actions

---An **algorithm** is the list of instructions and rules that a computer needs to do to complete a task.

The following list summarizes the key aspects :

- A computer is a tool that can be used to implement a plan for solving a problem.
- A computer program is a set of instructions for a computer. These instructions describe the steps that the computer must follow to implement a plan.
- An algorithm is a plan for solving a problem.
- A person must design an algorithm.
- A person must translate an algorithm into a computer program

Introduction to Algorithm Design

Why study Algorithm?

- We exposed to different problem-solving techniques and seeing how different algorithms are designed to solve a specific problems.

- we can learn analysis techniques that allow us to compare and contrast solutions based solely on their own characteristics, not the characteristics of the program or computer used to implement them.

- we may have a problem that is intractable, meaning that there is no algorithm that can solve the problem in a realistic amount of time. It is important to be able to distinguish between those problems that have solutions, those that do not.

- Finding a solution and then deciding whether it is a good one are tasks that we will do over and over again.

Introduction to Algorithm Design

What's the Importance of Algorithms in Computer Programming?

--- To create efficient and error free program

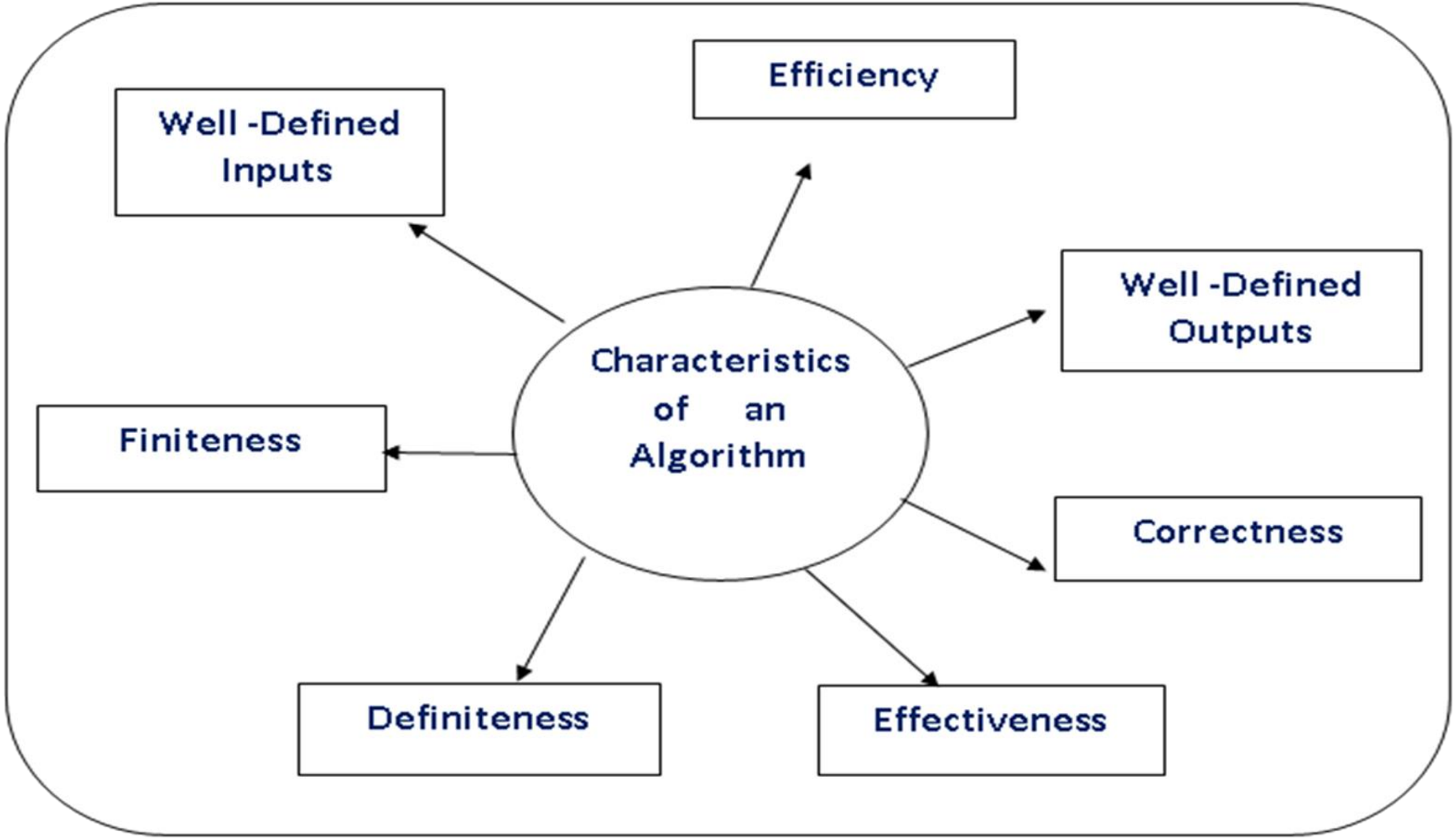
Speed/time, space, accuracy

---Proper utilization of resources (Right choice of algorithm ensures that a program consumes the least amount of memory)

--- Allows to break down the problems and provides conceptual solutions in term of steps.

--- As it has very important role in computer science ,it help us to get a good job in good company (help to get campus) and help to score a good mark in nationalize exams.

Introduction to Algorithm Design (Characteristic features of an algorithm)



Introduction to Algorithm Design (Characteristic features of an algorithm)

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 time.

An algorithm should have the following characteristics

1.Input :- An algorithm should have one or more well-defined inputs

2.Output :- An algorithm should have 1 or more well-defined outputs, and should match the desired output.

3. Finiteness :- It means that every algorithm should have finite number of steps. Or The process should be terminated after a finite number of steps.

4. Definiteness :-Each instruction of the algorithm should be clear and unambiguous.

5. Effectiveness :-Every instruction must be basic enough and should complete finite length of time.

6. Correctness :- It is said that the algorithm is correct with respect to a specification.

7. Efficiency :- Efficiency of an algorithm can be measured based on the usage of different resources.

Characteristic features of an algorithm

Definiteness

Example:

A program fragment is given as:

x ← 1, toss a coin,

if the result is head then x← 3

else x ← 4.

In the above program, all the steps would be carried out effectively but there is no definiteness since there are two possible values of x i.e., 1 and 3/4

Effectiveness:

An algorithm should be effective. Effective means that each step should be referred as principle and should be executing in finite time.

Example of Not Effectiveness: Find exact value of e using the following formula:

$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} \quad \ldots\ldots and\ add\ it\ to\ x$

It is not effective since it requires summation of infinite terms. Therefore, it takes infinite time hence not effective.

Characteristic features of an algorithm

Correctness:

How do we know whether an algorithm is actually correct?

First, the logical analysis of the problem we performed in order to design the algorithm should give us confidence that we have identified a valid procedure for finding a solution.

Second, we can test the algorithm by choosing different sets of input values, carrying out the algorithm, and checking to see if the resulting solution does, in fact, work.

BUT… no matter how much testing we do, unless there are only a finite number of possible input values for the algorithm to consider, testing can never prove that the algorithm produces correct results in all cases.

Third ,We can attempt to construct a formal, mathematical proof that, if the algorithm is given valid input values then the results obtained from the algorithm must be a solution to the problem.

Characteristic features of an algorithm

Efficiency:

Computer resources are limited that should be utilized efficiently. For maximum efficiency of algorithm we wish to minimize resource usage. The important resources such as time and space complexity could be considered for an algorithmic efficiency.

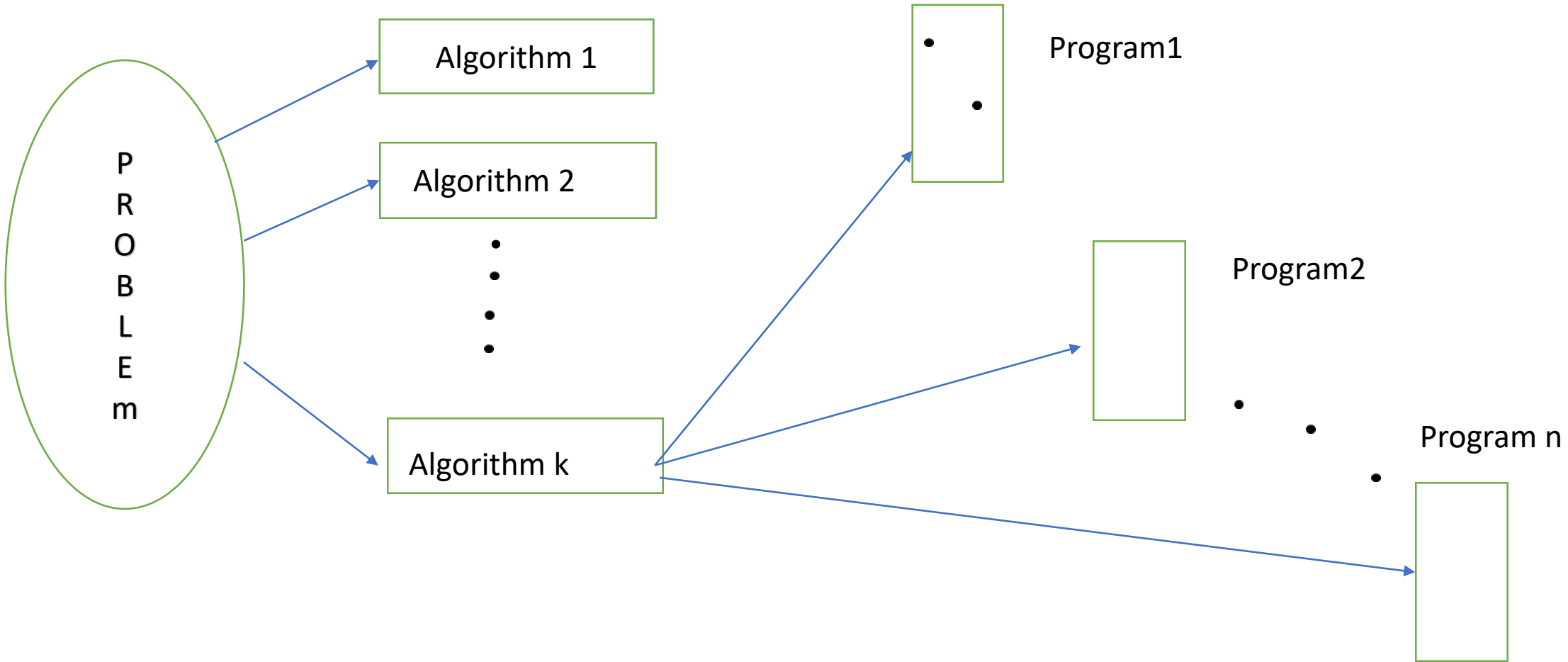
The time efficiency of an algorithm is measured by different factors. For example, write a program for a defined algorithm, execute it by using any programming language, and measure the total time it takes to run. The execution time that you measure in this case would depend on a number of factors such as:

- Speed of the machine,· Compiler and other system software tools
- Operating System ,· Programming language used
- Volume of data required

However, to determine how efficiently an algorithm solves a given problem, you would like to determine how the execution time is affected by the nature of the algorithm. Therefore, we need to develop fundamental laws that determine the efficiency of a program in terms of the nature of the underlying algorithm.

Problems vs Algorithms vs Programs

For each problem or class of problems, there may be different algorithms. For each algorithm, there may be different implementations (programs).



Expressing Algorithms

An algorithm may be expressed in a number of ways, including:

- natural language: Natural language is a popular choice, since it comes so naturally to us and can convey the steps of an algorithm.

2. However, natural language has its drawbacks. It has a tendency to be ambiguous since it has no imposed structure. That makes it difficult for others to follow the algorithm and feel confident in its correctness.

- flow charts: Flow charts and pseudocode are more structured formats that can more precisely express an algorithm .avoid most (if not all) issues of ambiguity; difficult to modify w/o specialized tools; largely standardized
- pseudo-code: most algorithms become code that actually runs on a computer. Before that happens, programmers often like to express an algorithm in code that uses all the constructs of a programming language, but doesn't actually run anywhere. also avoids most issues of ambiguity; vaguely resembles common elements of programming languages; no particular agreement on syntax
- programming language: tend to require expressing low-level details that are not necessary for a high-level understanding

Expressing Algorithms

Pseudo-code:

A means of describing an algorithm in human terms, without the use of a true programming **language**.

Pseudocode is an informal way of programming description that does not require any strict programming **language** syntax.

why do we need it?

- It is not a programming language.
- It is just a learning and reasoning tool, which is used by programmers and developers to underline how to write the actual code.
- Pseudocode can not be executed or compiled by any compiler, interpreter, or assembler.
- Unlike programming language code, pseudocode does not follow a strict structure and syntax. Here, the programmer can write the code syntax as he pleases.

Expressing Algorithms

- How to Write Pseudocode?

As pseudocode does not follow a strict systematic or standard way of being written, so don't think of writing pseudocode as a strict rule. However, there are some standard conventions that every programmer follows while writing one. These are:

- Use capital words for reserved commands or keywords, for example, if you are writing IF…ELSE statements then make sure IF and ELSE be in capital letters.
- Write only one statement per line.
- Use indentation for the block body. It keeps the body of every component isolated and indenting different pieces of each block will indicate that those pieces of pseudocode go under a less intended section.
- Be specific while writing a statement, use plain English to provide a particular description.

Expressing Algorithms

Pseudocode conventions for different programming statements

1. Assignment Operator: =, <- or := . Variable := expression a:=0 a<-0 a=0
2. Comparison Operator: == , !=, <, >, <= , and >=
3. Arithmetic Operator: +,-, *, /, mod(%)
4. Logical Operator: and, not and or
5. Sum, Product : Σ Π
6. Comment line begins with // and continue until the end of line
7. Block are indicated with matching braces { and }
8. An identifier begin with a letter
9. Elements of multidimensional array are accessed using [and]. If A is two dimensional array then array is denoted by A[i,j] and array indices start at zero

Expressing Algorithms

Conditional Statements:

if	if…else	if…else if….else
if (condition) then if body	**if** (condition) then if body **else** else body	**if** (condition) then if body **else if** condition then else if statement **else** else body

age = input “Enter your age”
If (age <= 5)
print “infants/kids”
	age = input : “Enter your age”
	If (age >=18)
	print “adult”
	else
	print “Under age”
		age = input : “Enter Your age”
		
		If(age ==18)
		then print “under check”
		else if (age > 18)
		print “Give entry”
		else
		print “under age”

Expressing Algorithms

Case statement:

```
case <expression>
{
   constant 1:statement 1
    …….
     …….
     constant n: statement n
}
```

Expressing Algorithms

Iterators:

for loops	while loop

for variable:= value1 to value2 **do**
{
for body
}
	while (condition) **do**
	{
	while body
	}

for i := 0 to 20 do
{
print i
}
	i := 0
	while(i <= 20)
	{
	print i
	}

Function:

```
function name(parameters)
{
   function body
   return (value);
}
```

```
add( para1, para2 )
{     result:=para1+para2;
      return(result);
}
```