Ways of Writing an Algorithm

There are various ways of writing an algorithm.

1. English-Like Algorithm

Algorithm can be written in many ways. 1st method is Simple English Language.

Eg :

Algorithm : English – like algorithm of linear search.

Step 1: Compare `item` with the first element of the array, A.

Step 2: If the two are same, then print the position of the element and exit.

Step 3: Else repeat the above process with the rest of the elements.

Step 4: If the item is not found at any position, then print 'not found' and exit.

Disadvantage:

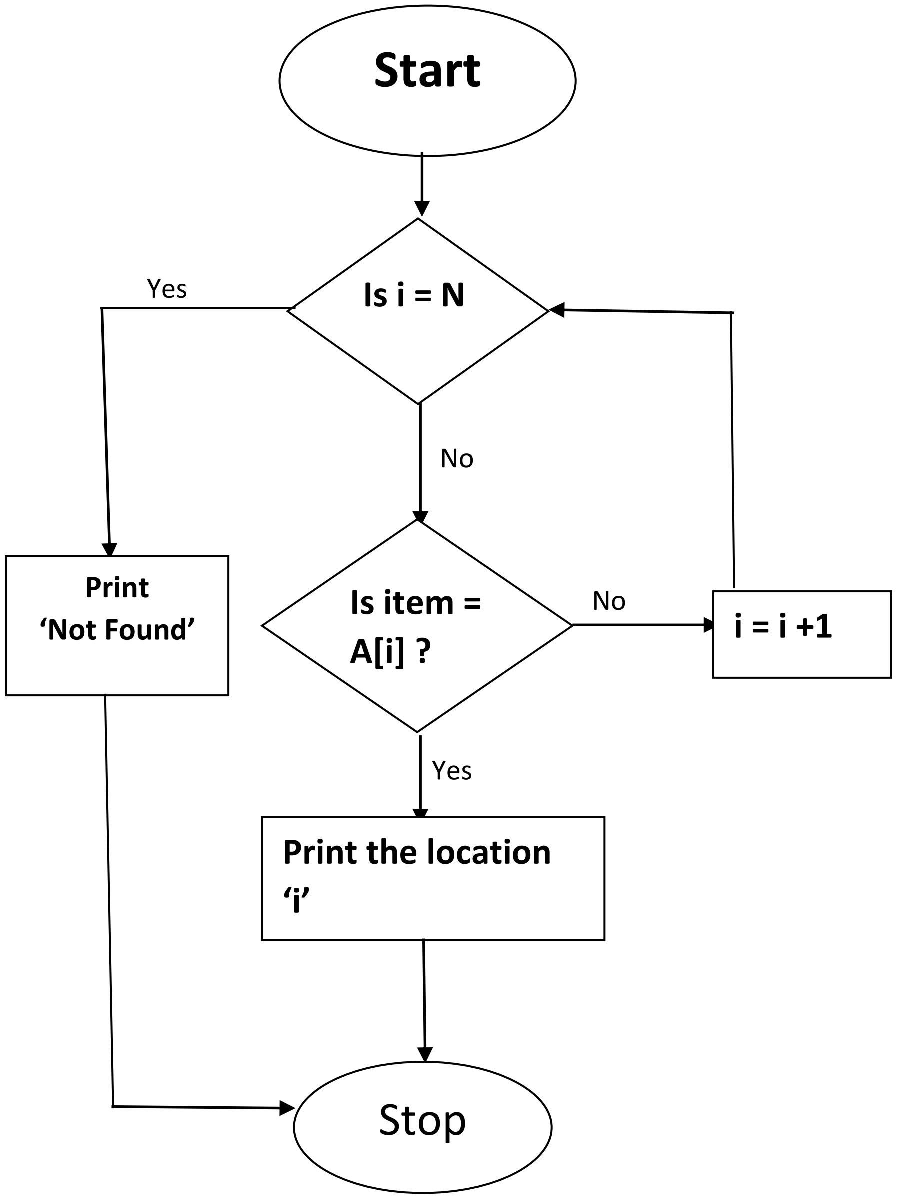
1. Natural Languages can be ambiguous and therefore its lack the characteristics of being definite.

2. English language-like algorithms are not considered good for most of the task.

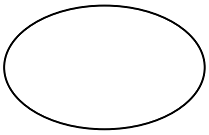
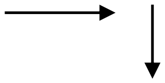
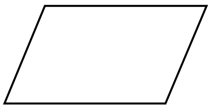
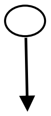
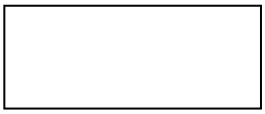
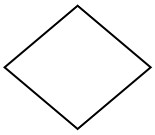
2. Flow Chart

Flowcharts pictorially depict a process.

They are easy to understand and are commonly used in the case of simple problems.



Flow Chart Conventions

S. No.	Name	Element Representation	Meaning
1.	Start/End		An oval is used to indicate the beginning and end of an algorithm.
2.	Arrows		An arrow indicates the direction of flow of the algorithm.
3.	Input/Output		A parallelogram indicates the input or output.
4.	Connectors		Circles with arrows connect the disconnected flowchart.
5.	Process		A rectangle indicates a computation.
6.	Decision		A diamond indicates a point where a decision is made.

3. Pseudocode

The pseudocode has an advantage of being easily converted into any programming . This way of writing algorithm is most acceptable and most widely used.

Pseudocode Conventions

S. No.	Construct	Meaning
1.	//	Single Line Comment
2.	/* Comment Line 1 Comment Line 2 . . Comment Line n */	Multi-line comments occur between /* … */

3.	{
	Statements
	}
		Blocks are represented using { and } . Blocks can be used to represent compound statements(collections of simple statements) or the procedure.

4.	;	Statements are delimited by ;
5.	<variable> = <Expression>	This is an assignment statement. The statement indicates that the result of evaluation of expression will be stored in the variable.
6.	a> b	a and b are expressions, and > is a relational

		operator ‘greater than’ . The Boolean expression a > b returns true if a is greater than b , else returns false.
7.	a < b	a and b are expressions, and < is a relational operator ‘less than’ . The Boolean expression a < b returns true if a is less than b , else returns false.
8.	a <= b	a and b are expressions, and < is a relational operator ‘less than or equal to’ . The Boolean expression a < b returns true if a is less than or equal to b , else returns false.
9.	a>=b	a and b are expressions, and > is a relational operator ‘greater than or equal to’ . The Boolean expression a >= b returns true if a is greater than or equal to b , else returns false.
10.	a! = b	a and b are expressions , and != is a relational operator ‘not equal to’ . The Boolean expression

		a!=b returns true if a is not equal to b, else returns false.
11.	a == b	a and b are expressions, and == is a relational operator ‘equal to’. The Boolean expression a == b returns true if both a is equal to b, else returns false.
12.	a AND b	a and b are expressions, and AND is logical operator. The Boolean expression a AND b returns true if both the conditions are true, else it returns false.
13.	a OR b	a and b are expressions, and OR is logical operator. The Boolean expression a OR b returns true if any of the condition is true, else it returns false.
14.	NOT a	a is an expression, and NOT is a logical operator. The Boolean expression ‘NOT a’ returns true if the result of ‘a’ evaluates to False,

		else returns False.
15.	If<condition>then: <statement>	The statement indicates the conditional operator if.
16.	If<condition> then : <statement1> Else: < statement2>	The statement is an enhancement of the above if statement. It can also handle the case wherein the condition is not satisfied.

17.	Case
	{
	:<condition1>: <statement1>
	.
	.
	:<condition n>: <statement n>
	:default: <statement n+1>
	}
		The statement is a depiction of switch case used in C or C++.

18.	While<condition>do
	{
	statements
	}
		The statement depicts a while loop.

19.	repeat
	statements
	until<condition>
		The statement depicts a do-while loop.

20.	for variable = value1 to value2
	{
		The statement depicts a for loop.

	statements }	
21.	Read	Input instruction
22.	Print	Output instruction
23.	Algorithm<name>(<parameter list>)	The name of the algorithm is <name> and the arguments are stored in the <parameter list>

Eg : Algorithm : Linear Search on basis of Pseudocode.

```
Algorithm Linear_Search(A,n,item)
{
for i = 1 to n step 1 do
   {
     If(A[i] == item)
     {
       print i;
        exit();
```

```
      }
    }
   print “Not Found”
}
```