

Data structure operations

Traversing : Traversing means accessing each record (element) only once so that it can be processed.

Searching: Searching means finding the location of record (element) with given by value or finding all records that satisfies condition.

Inserting : Inserting means adding new record to the structure.

Deleting : Deleting means removing the record (element) from the structure.

Sorting: Sorting means arranging elements in some logical order.

Merging : Merging means combining the elements of two different files into a single file.

Algorithmic notation

An algorithm is a finite step-by-step list of well defined instructions for solving a particular problem.

Algorithm consists of two parts. The first part of algorithm tell the purpose of algorithm. It lists for variables & input data. The second part is a list of steps.

The steps in algorithm are executed one after the other. Control may be transferred to step n by using statement like 'Go to step n'.

The exit and stop statements completes the algorithm.

The data may be assigned to variables by using read statement and it is displayed by write or print statement.

Control Structures

There are three types of flow of control (or logic) :

Sequential flow (Sequential logic) - In sequential flow the modules are executed one after the other.

Module A

Module B

Module C

Conditional flow: In conditional flow, one or other module is selected depending on condition. There are three conditional flow-

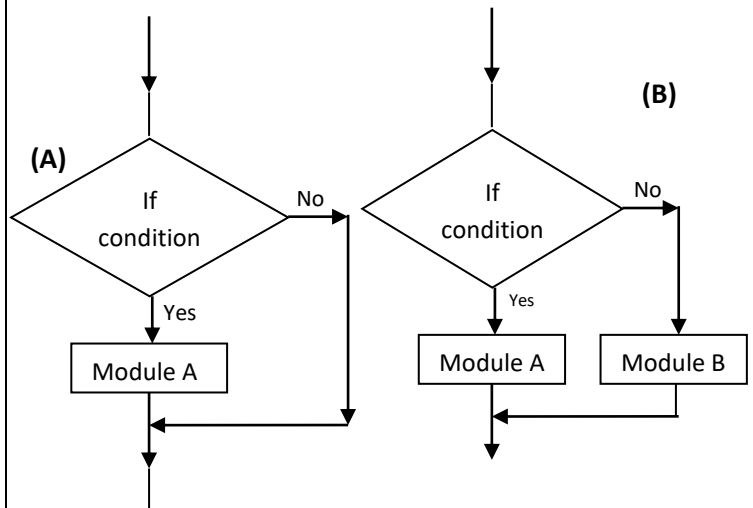
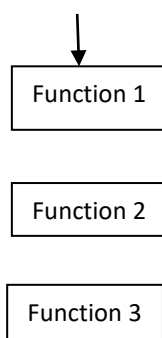
A. Single alternative

This has the form -

If condition, then:

[Module A]

[End of IF structure]



B. Double alternative

This structure has the form –

If condition, then :

[Module A]

ELSE :

[Module B]

[End of IF structure]

C. Multiple alternatives - The logic of this structure allows only one module to be executed.

This structure has the form –

If condition (1), then :

[Module A1]

ELSE if condition (2), then :

Module A2]

ELSE if condition (M), then :

[Module AM]

ELSE :

[Module B]

[End of IF structure]

Repetitive flow: Here

certain module is executed repeatedly unity condition satisfies.

The repeat for loop has the from :

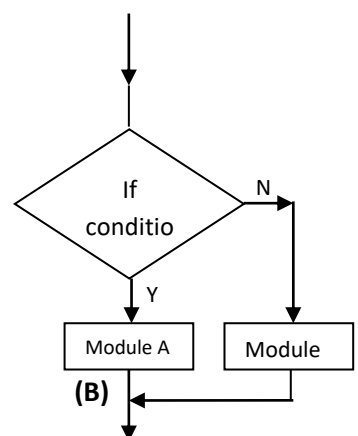
Repeat for K = R to S by T

[Module]

[End of if loop]

Here K is index variable.

Initial value of K is R and final value is S. T is increment.



Algorithms

1. Bubble sort.

Bubble (DATA, N)

Here DATA is an array having N elements. This algorithm sorts the elements in array.

1. Repeat steps 2 and 3 for I= 1 to N - 1
2. Set PTR = 1
3. Repeat while PTR <= N - K:
 - (a) If DATA [PTR] > DATA [PTR + 1] then:

Interchange DATA [PTR] and DATA [PTR + 1]

[End of IF structure]
 - (b) Set PTR = PTR + 1

[End of inner loop]

[End of step 1 outer loop]
4. EXIT.

2. Linear search

LINEAR (DATA, N, V, LOC)

Here DATA is a linear array with N elements, and V is given element which we have to search. This algorithm finds the location (LOC) of V in DATA, or sets LOC = 0 if search is unsuccessful.

1. Insert V at the end of array.

Set DATA [N + 1] = V
2. Initialize counter.

Set LOC = 1
3. Search for item

Repeat while DATA [LOC] != ITEM :

Set LOC = LOC + 1

[End of loop]
4. If LOC = N + 1, then set LOC = 0
5. EXIT.

3. Binary search

Binary (DATA, LL, UL, V, LOC)

Here DATA is sorted array with lower limit LL and upper limit UL. V is given element which is to be searched. BEG denotes beginning, MID denotes middle and END denotes end locations of segment of ARRAY. This algorithm finds the location LOC of V in ARRAY or set LOC = NULL if search is unsuccessful.

1. Set BEG = LL, END = UL and MID = INT ((BEG + END) / 2)
2. Repeat steps 3 and 4 while BEG < END and DATA [MID] != V.
3. IF V < DATA [MID], then:

Set END = MID - 1.

EISE :

Set BEG = MID + 1

[End of IF structure]

4. Set MID = INT ((BEG + END) / 2)

[End of step 2 loop]

5. If DATA [MID] = V, then:

Set LOC =MID

ELSE :

Set LOC = NULL

[End of IF structure]

6. EXIT.

4. Inserting element into array

INSERT (DATA, N, K, V)

Here DATA is linear array with N elements and K is positive integer such that $K \leq N$. This algorithm inserts an element V into the K^{th} position in array.

1. Initialize counter.

Set J = N
2. Repeat steps 3 and 4 while J? K.
3. Move J^{th} element downward.

Set DATA [J + 1] = DATA [J]
4. Decrease the counter.

Set J=J-1.
5. Insert the element

Set DATA [K] = V
6. Reset N.

N = N + 1.
7. EXIT.

5. Inserting element into Array

DELETE (DATA, N, K, V)

Here DATA is linear array with N elements and K is a positive integer such that $K \leq N$. This algorithm deletes the K^{th} element from array.

1. Set V= DATA [K]
2. Repeat for J = K to N - 1


Move J + 1st element upward.


Set DATA [J] = DATA [J + 1]


[End of loop]
3. Reset N


N = N - 1.
4. EXIT.


Stack

 A Stack is a data structure in which items may be added or removed only at one end.

 The common examples of stack are stack of dishes, a stack of books etc.

 The item can be removed or added only from the top of the stack.

 "Push" is the term used to insert an element into a stack. "Pop" is the term used to delete an element from a stack.

 A stack is also called as last-in-first-out (LIFO) list.

Queue

👉 A queue is a linear list in which items may be added only at one end and items may be removed only at other end.

👉 The common example of queue is the queue waiting for a bus at bus stop.

👉 The queue is also called as first-in-first-out (FIFO) list.

Linked lists

👉 A linked list is a linear collection of ARRAY elements, called nodes, where the linear order is given by pointers.

👉 Each node is divided into two parts. The first part contains the information of the element and second part is link field which contains the address of the next node in the list.

👉 In linked lists, insertion and deletion is easy. Linked list also do not require consecutive memory locations

When list do not have any node then it is called null list or empty list.