Studoob.in - Where Learning is Entertainment

Studoob.in - Where Learning is Entertainment

Data structure is the way of organising similar or dissimilar logically related data items.

Data structure

- Concrete
- Abstract

again classified to
- Premitive - basic unit - integer, char
- Non premitive → array
  - can divide as basic units

static data structure → at the time of compile memory is allocated - array

dynamic d.s → at the time of execution memory is allocated - linked list

Linear ⇒ stored one by one → stack, array, queue

nonlinear ⇒ tree, graph

continuous → array noncontinuous → linked list

Module - 1

1. write an algorithm to insert an element at the beginning of the array.

A.

1. Initialize maximum no. of elements to be stored, MAX. 5
2. Input no. of elements of an array, n. 4
3. Input elements of the array A 2 3 4 5
4. Input the element to be inserted, K = 1 max = 5, n = 4
5. i = n i = 4
6. if n = MAX
   print "Insertion is not possible"
   else
   while i ≥ 1
   {
   A[i+1] = A[i]
   i = i - 1
   }
7. A[i] = K
8. n = n + 1
9. print elements of the array A.

Scanned by CamScanner

Studoob.in - Where Learning is Entertainment
Studoob.in - Where Learning is Entertainment

2. write an algorithm to insert an element at the end of an array.

A. 1. Initialize maximum no:of elements to be Stored, max.
2. Input the no:of elements of the array, n
3. Input the elements of the array A
4. Input the number to be inserted, k
5. if n = max

    print "Insertion is not possible"

else

    A[n+1] = K

6. n = n+1
7. print elements of the array

3. write algorithm to insert an element at the particular position.

1. Initialize maximum no:of elements to be stored, max
2. Input the no:of elements of the array, n
3. Input the elements of the array A
4. Input the number to be inserted, k
5. input position, p.
6. i = n
7. If n = max

    print "Insertion is not possible"

else

    while i ≥ P
    {
        A[i+1] = A[i]
        i = i - 1
    }

8. A[P] = K
9. n = n+1
10. print the elements of the array.

4. write algorithm to delete an element i) from the beginning.
ii) from end
iii) from particular position.
and print the deleted element

A. i)
1. Initialize maximum number of elements of the array, max
2. Input the no:of elements of the array, n.
3. Input the elements of the array A
4. i = 0
5. if n = 0

    print "deletion is not possible"

else

    while i ≤ n
    {
        A[i-1] = A[i]
        i = i+1
    }

This is also i

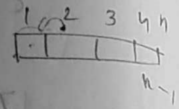
max = 5

i = 2

2	20
3	30
4	40

Studoob.in - Where Learning is Entertainment
Studoob.in - Where Learning is Entertainment

6. ~~n = n - 1~~
7. ~~print A[~~



4. print A[i]
5. i = 1
6. if n = 0
   print "Deletion is not possible"
   else
   while i<n
   {
   A[i] = A[i+1]
   i = i+1
   }
7. n = n-1
8. print elements of the array.

ii)

1. Initialize maximum no:of elements of the array, max
2. Input the number of elements, n
3. Input all elements of the array A
4. print A[n]
5. If n = 0
   print "deletion is not possible"

else
   n = n-1
6. print elements of the array.

iii)

1. Initialise maximum no:of elements to be stored, max.
2. Input no:of elements, n
3. Input the elements of array A
4. Input the particular position to be deleted, m
5. if n=0
   print "deletion is not possible
   else
   while (i ≥ m)
   {
   A[i] = A[i+1]
   i = i+1
   }
6. n = n-1
7. print the elements of the array, A

Scanned by CamScanner

Studoob.in - Where Learning is Entertainment
Studoob.in - Where Learning is Entertainment

5. Write algorithm to find the no: of occurences of the given element in an array of size n.

Representation of Arrays In Memory

There are two types of representation of array in memory

i) row major - stored in rows
ii) column major - " " columns

How to calculate the address of a particular element

Suppose the order of a 2D array is $m \times n$. What is the address of A[i][j]. Assume that the base address is B.

A* In row major method

For 2D array,

eqn is $\boxed{(i-1) \times n + j}$ for n = no: of columns

(index = 1)

$= B + (i-1) \times n + j$

* In column major method,

eqn is $\boxed{(j-1) \times m + i}$

$= B + (j-1) \times m + i$

5.A. 1. Initialise flag = 0, a[20]
2. Input the no: of elements n
3. Input the array elements
4. Input the searching element, k
5. for (i=0; i<=n; i++)
   if a[i] == k)
   flag ++
6. print numbers "k is occuring repeated/ as __ times"

3 dimensional array

A[i, j, k]
m, n, p

Row major :-

$(i-1) \times n \times p + (j-1) \times p + k$

For n dimensional array:

$m_1, m_2, m_3, \dots, m_n$

$A[i_1, i_2, i_3, \dots, i_n]$

$((i_1-1) \times m_2 \times m_3 \times \dots \times m_n +$

$(i_2-1) \times m_3 \times m_4 \times \dots \times m_n +$

$\dots\ i_{n-1} - 1 \times m_n + i_n)$

Studoob.in - Where Learning is Entertainment
Studoob.in - Where Learning is Entertainment

Sparse matrix

It is the matrix in which majority elements are zero.

In sparse matrix representation no: of columns is limited to 3 columns (index starts from zero),

then the number of rows = no: of non zero element + 1.

row	position	element

In first row,

⟹

Total no: of rows	total no: of columns	total no: of non zero ele

For example :-

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 10 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 \end{bmatrix} \Rightarrow \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{bmatrix} 4 & 4 & 3 \\ 0 & 2 & 1 \\ 2 & 0 & 10 \\ 3 & 1 & 20 \end{bmatrix}$$

Linked list

Linked list is used for dynamic memory allocation.

Scanned by CamScanner

Studoob.in - Where Learning is Entertainment
Studoob.in - Where Learning is Entertainment

6. write algorithm and program to search an element from a set of n elements using binary search.

A. 1. Input the no: of elements of the array, n

2. Input the elements of array A

3. Sort the array.
4. input searching element, k
5. initialise i=0 and j = n-1
6. while i <= j

```
{
  mid = (i+j)/2

  if A[mid] = K
     print "found"
     break
  else {
  if A[mid] < K
       i = mid+1
  else:
       j = mid - 1
  }
}
```

7. if i > j
   print "not found!!"

OR

1. Input the no: of elements, n
2. Input the element of array A
3. sort the array.
4. Input searching element, K
5. Initialise i = 0 and j = n-1, flag = 0
6. while i<=j

```
{
   mid = (i+j)/2

   if A[mid] = K
      print "Found" and flag = 1
      break
   else
      if A[mid] < K
          i = mid + 1
      else
          j = mid - 1
}
```

7. if flag = 0
   print "not found!!!"

Linear Sorting

1. Input the numbers of elements of array, n
2. Input the element of array A
3. for (i = 0 ; i ≤ n - 2 ; i++)

```
   for (j = i+1 ; j < n-1 ; j++)
        if A[i] > A[j]
            temp = a[i]
```

Scanned by CamScanner