Unit-II
Data Structure

Arrays, Stacks, Queues And Linked List — Chapter: 06

In Computer Science, a **data structure** is a particular way of storing and organizing data in a computer so that it can be used efficiently. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks.

Simple Data Structure: These data structures are normally built from primitive data types like integers, floats, characters. For example arrays and structure.

Compound Data Structure: simple data structures can be combined in various ways to form more complex structure called compound structures. Linked Lists, Stack, Queues and Trees are examples of compound data structure.

Searching methods in array

Linear Search: In this method each element of the array is compared with the number to be searched in linear order (from first to last). And where the number is matched the position is displayed.

```
#include<iostream.h>
#include<conio.h>
void main()
{
 int lsearch(int[],int,int);
 int a[50],item,n,index;
 clrscr();
 cout<<"\n Enter size of array";
 cin>>n;
 cout<<"\n Enter array elements";
 for(int i=0;i<n;i++)
 cin>>a[i];
 cout<<"Enter the item to be searched";
 cin>>item;
 index=lsearch(a,n,item);
 if(index= = -1)
 cout<<"\n Element not found";
 else
 cout<<"\n Element found at position
"<<index+1;
 getch();
 }

 int lsearch(int a[],int size,int item)
 {
  for(int i=0;i<size;i++)
  {
```

Binary Search Method

Binary search algorithm is applicable for already sorted array only. In this algorithm, to search for the given item from the sorted array (in ascending order), the item is compared with the middle element of the array. If the middle element is equal to the item then index of the middle element is returned, otherwise, if item is less than the middle item then the item is present in first half segment of the array (i.e. between 0 to middle-1), so the next iteration will continue for first half only, if the item is larger than the middle element then the item is present in second half of the array (i.e. between middle+1 to size-1), so the next iteration will continue for second half segment of the array only. The same process continues until either the item is found (search successful) or the segment is reduced to the single element and still the item is not found (search unsuccessful).

```
#include<iostream.h>
#include<conio.h>

void main()
{
int bsearch(int[],int,int);
 int a[50], item, n, index;
 clrscr();
 cout<<"\n Enter total elements";
 cin>>n;
```

```
  if(a[i]==item)
  return i;
}
return -1;
  }
```
	```
	 cout<<"\n Enter array elements in sorted
	form:";
	 for(int i=0;i<n;i++)
	 cin>>a[i];
	 cout<<"Enter the item to be searched";
	 cin>>item;
	 index=bsearch(a, n, item);
	 if(index= = -1)
	 cout<<"\n Element not found";
	 else
	 cout<<"\n Element found at position
	"<<index+1;
	 getch();
	 }

	 int bsearch(int a[], int size, int item)
	{
	 int beg, last;
	 beg=0; last=size-1;
	 int mid=(last+beg)/2;
	 while(beg<=last)
	 {
	  mid=(beg+last)/2;
	  if(item= =a[mid])
	          {
	return mid;
	}
	  else if(item>a[mid])
	beg=mid+1;
	  else
	last=mid-1;
	  }
	  return -1;
	  }
	```

Sorting operation in the array

Sorting means to arrange the array elements in Ascending order or in Descending order. There are various methods to do this but for the ease sake Bubble sort method is displayed here.

```
   #include<iostream.h>
 #include<conio.h>
 void bubblesort (int[],int);
 void main()
  {
   int a[50],n;
   clrscr();
   cout<<"\nHow many elements do you want to create array with? ";
   cin>>n;
   cout<<"\nEnter array elements\n";
  for(int i=0;i<n;i++)
   cin>>a[i];
  bubblesort(a,n);
  cout<<"\n\nThe sorted array is as shown below\n";
  for(i=0;i<n;i++)
  cout<<a[i]<<"\n";
  getch();
 }
 void bubblesort(int a[],int n)  //Function to perform bubble sort
  {
   int temp;
   for(int i=0;i<n-1;i++)
    {
        for(int j=0;j<n-i-1;j++)
         if(a[j]>a[j+1])
          {
           temp=a[j];
           a[j]=a[j+1];
           a[j+1]=temp;
          }
    }
  }
```

Some Questions Based on Array (2 marks question)

Q1. Write a function in C++ which accepts an integer array and its size as arguments/parameters and reverses the array
example : if the array is 1,2,3,4,5 then rearrange the array as 5,4,3,2,1
Ans : void reverse(int arr[ ], int n)

```
{
        int temp;
for(int i=0,j=n-1; i<=j; i++,j--)
        {
        temp= arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
        }
}
```

Q2. Write a function in C++ which accepts an integer array and its size as arguments/parameters and exchange the array in the given manner
example : if the array is 1,2,3,4,5,6,7,8,9,10 then rearrange the array as2,1,4,3,6,5,8,7,10,9
Ans : void change(int arr[ ], int n)

```
{
        int temp;
for(int i=0; i<n; i=i+2)
        {
        temp= arr[i];
        arr[i] = arr[i+1];
        arr[i+1] = temp;
}
}
```

Q3 Write a function in C++ to merge the contents of two sorted arrays A & B into third array C. Assuming array A is sorted in ascending order, B is sorted in descending order, the resultant array is required to be in ascending order.

Q 4 Write a function in C++ which accepts an integer array and its size as arguments and assign the elements into a two dimensional array of integers in the following format

If the array is 1,2,3,4,5,6	if the array is 1,2,3
The resultant 2D array is	The resultant 2D array is
1 2 3 4 5 6	1 2 3
0 1 2 3 4 5	0 1 2
0 0 1 2 3 4	0 0 1
0 0 0 1 2 3	
0 0 0 0 1 2	
0 0 0 0 0 1	

Questions based on Two Dimensional Array(3 marks question)

Q1. Write a function in C++ that will accept a 2-D array and its row and column size as argument and find sum of rows and columns

Ans : void rowcolsum(int A[ ][ ],int N, int M)

```
{
        for (int i=0;i<N;i++)
        {
                int SumR=0;
                for (int j=0;j<M;j++)
                        SumR+=A[i][j];
                cout<<SumR<<endl;
        }
        for (int i=0;i<N;i++)
        {
                int SumC=0;
                for (int j=0;j<M;j++)
                        SumC+=A[j][i];
                cout<<SumC<<endl;
        }
  }
```

Q2. Write a function in C++ to find the sum of both left and right diagonal elements from a two dimensional array (matrix).

Ans : void DiagSum(int A[ ][ ], int N)

```
{
int SumD1=0,SumD2=0;
for (int I=0;I<N;I++)
{
        SumD1+=A[I][I];
```

SumD2+=A[N-I-1][I];
}
cout<<"Sum of Diagonal 1:"<<SumD1<<endl;
cout<<"Sum of Diagonal 2:"<<SumD2<<endl;
}

Address Calculation in Two Dimensional Array

Two dimensional array can be arranged in two manner

1. Row Major Order
2. Column Major Order

To find the address of a particular row and column the formula in **Row Major Order**is

Address of A[row][column]=B +w*(n(row)+column)

Where

B= Base address of the array

w= Word size

n= total no of columns in the array

To find the address of a particular row and column the formula in **Column Major Order** is

Address of A[row][column]=B +w*(m(Column)+row)

Where

B= Base address of the array

w= Word size

m= total no of rows in the array

Q1. An array x[30][10] is stored in the memory with each element requiring 4 bytes of storage. If the base address of x is 4500, find out memory locations of x[12][8] if the content is stored along the row.

Ans: Here the array is stored in Row Major Order so

B=4500

W= 4

N= 10

As per the formula

Address of A[row][column]=B +w*(n(row)+column)

=4500+4*(10(12)+8)

=4500+4*(128)

=4500+512

=5012

Q 2. An array P[20][30] is stored in the memory along the column with each of the element occupying 4 bytes, find out the Base Address of the array, if an element P[2][20] is stored at the memory location 5000.

Ans : Given, W=4, N=20, M=30, Loc(P[2][20])=5000

Column Major Formula:

Loc(P[I][J]) =Base(P)+W*(N*J+I)

Loc(P[2][20]) =Base(P)+4*(20*20+2)

Base(P) =5000 – 4*(400+2)

=5000 – 1608

=3392

Q3. An array S[40][30] is stored in the memory along the row with each of the element occupying 2 bytes, find out the memory location for the element S[20][10], if an element S[15][5] is stored at the memory location 5500.

Ans. Given, W=2, N=40, M=30, Loc(S[15][5])=5500

Row Major Formula:

Loc(S[I][J]) =Base(S)+W*(M*I+J)

Loc(S[15][5]) =Base(S)+2*(30*15+5)

5500 =Base(S) + 2*(450+5)

Base(S) =5500 – 910 = 4590

Loc(S[20][10]) =4590+2*(30*20+10)

=4590+2*(600+10)

=4590+1220 = 5810