



THE UNIVERSITY
OF LAHORE
**ISLAMABAD
CAMPUS**

**Data Structures and Algorithms
(CS09203)**

Lab Report

Name: Rehman ullah baig
Reg No: CSU-F16-118
Lab Report: 1
Submitted To: Sir, Usman Ahmed

Experiment # 01

Introduction to Arrays and its operation

Objectives:-

The objectives of this lab session are to understand the basic and various operations on arrays in C++.

Software Tools:-

- Window 7
- Dev C++
- Ms office 2003

Theory:-

We have already studied array in our computer programming course. We would be using the knowledge we learned there to implement different operation on arrays.

Traversing Linear Arrays:-

Let A be the collection of data elements stored in the memory of the computer. Suppose we want to print the contents of each element of A or suppose we want to count the number of elements of A with a given property. This can be accomplished by traversing A that is by accessing and Processing each element of A exactly once.

The following algorithm traverses a linear array. The simplicity of the algorithm comes from the fact that LA is a linear structure. Other linear structures such as linked list can also be easily traversed. On the other hand the traversal of non-linear structures such as trees and graphs is considerably more complicated.

Algorithm:-

(Traversing a Linear Array) Here LA is a linear Array with lower Bound LB and upper Bound UB. This algorithm traverses LA.

Applying an operation PROCESS to each element of LA.

[Initialize Counter] Set $X=LB$.
1. Repeat Step 3 and 4 while
 $K \leq UB$. [Visit element] Apply
PROCESS to $LA[X]$. [Increase
Counter] Set $X=X+1$.
[End of Step 2 Loop]
5. Exit.

Inserting and Deleting: -

Let A be a collection of data elements in the memory of computer. “Inserting” refers to the operation of adding another element to the collection A and “deleting” refers to

the operation of removing one of the elements from A. Here we discuss the inserting and deleting when A is a linear array.

Inserting an element at the “end” of the linear array can be easily done provided the memory space allocated for the array is large enough to accommodate the additional element. On the other hand suppose we need to insert an element in the middle of the array. Then on average half of the elements must be moved downward to the new location to accommodate the new element and keep the order of other elements.

Similarly deleting the element at the “end” of an array presents no difficulties but deleting the element somewhere in the middle of the array would require that each subsequent element be moved one location upward in order to fill up the array.

Algorithm of Insertion operation: -


(Inserting into Linear Array) INSERT (LA, N, K, ITEM)

Here LA is a linear array with N elements and K is a positive integer such that $K \leq N$. This algorithm inserts an element ITEM into the Kth position in LA.

1. *[Initialize Counter] Set $J=N$.*
2. *Repeat Step 3 and 4 while $J \geq K$.*
3. *[Move Jth element downward] Set $LA [J+1] = LA[J]$.*
4. *[Decrease Counter] Set $J=J-1$.*
 End of Step 2 Loop.
5. *[Insert element] Set $LA[K]=ITEM$.*
6. *[Reset N] Set $N=N+1$.*
7. *Exit.*

Lab Task: -

Write a C++ program to implement all the above described algorithms and display the following menu and ask the user for the desired operation.



```
C:\Users\Masti Malkal\Documents\dd.exe
Enter values:
7
8
9
5
6
Enter a number to find:
6
6 is on location 4
-----
```

```
C:\Users\Masti Malkal\Documents\lab 1 q 2.exe
Enter the Values:
1
2
3
4
5
Enter the value to insert:
6
Enter position where value has to be insert:
5
1
2
3
4
5
6
_____
```

```
C:\Users\Masti Malkal\Documents\deletion.exe
Enter the values of array:
1
2
3
4
5
Stored elements in array:
1
2
3
4
5
Enter position to delete:
3
New Data in Array:
1
2
4
5
_____
```

Conclusion

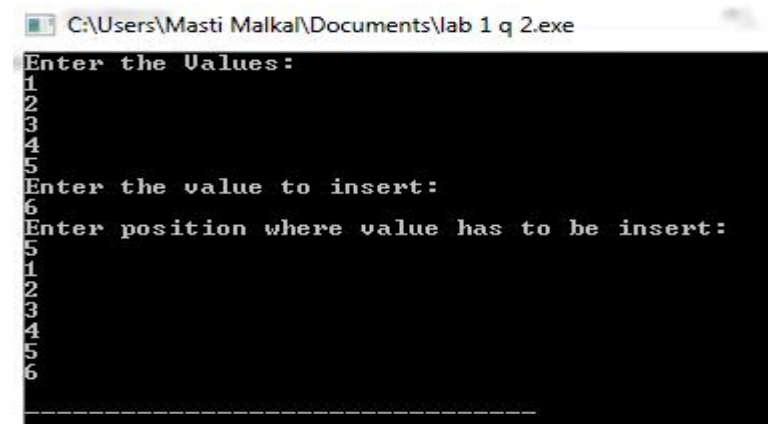
Insertion:

Code:

```
#include<iostream>
using namespace std;
int main()
{
    int arr[10], i,j, k;

    cout<<"Enter the Values:"<<endl;
    {
        for(i=0; i<5; i++)
            cin>>arr[i];
    }
    int ins,pos;
    cout<<"Enter the value to insert:"<<endl;
    cin>>ins;
    cout<<"Enter position where value has to be insert:"<<endl;
    cin>>pos;
    for(j=4; j>=pos; j--)
    {
        arr[j+1] = arr[j];
    }
    arr[pos] = ins;
    for(k=0; k<6; k++)
    {
        cout<<arr[k]<<" "<<endl;
    }
    return 0;
}
```

Output:



```
C:\Users\Masti Malka\Documents\lab 1 q 2.exe
Enter the Values:
1
2
3
4
5
Enter the value to insert:
6
Enter position where value has to be insert:
5
1
2
3
4
5
6
```

Deletion:

Code:

```
#include<iostream>
using namespace std;
int main()
{
    int i,A[5];
    cout<<"Enter the values of array:"<<endl;
```

```

    {
        for(i=0; i<5; i++)
            cin>>A[i];
    }
    cout<<"Stored elements in array:"<<endl;

    for(i=0; i<5; i++)
    {
        cout<<A[i]<<" "<<endl;
    }
    int pos;
    cout<<"Enter position to delete:"<<endl;
    cin>>pos;

    --pos;

    for(i=pos; i<=4; i++)
    {
        A[i]=A[i+1];
    }
    cout<<"New Data in Array:"<<endl;

    for(i=0; i<4; i++)
    {
        cout<<A[i]<<" "<<endl;
    }
    return 0;
}

```

Output:

```

C:\Users\Masti Malkal\Documents\deletion.exe
Enter the values of array:
1
2
3
4
5
Stored elements in array:
1
2
3
4
5
Enter position to delete:
3
New Data in Array:
1
2
4
5

```

Traversing:

Code:

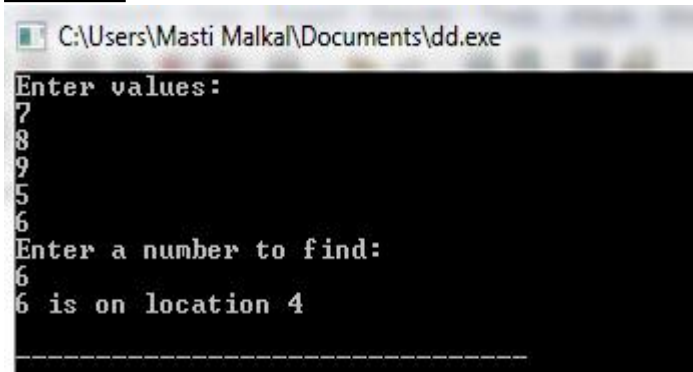
```

#include<iostream>
using namespace std;
int main()
{
    int arr[5];

```

```
int num;
cout<<"Enter values:"<<endl;
{
    for(int i=0; i<5;i++)
        cin>>arr[i];
}
cout<<"Enter a number to find:"<<endl;
cin>>num;
for(int j=0; j<5; j++)
{
    if(num==arr[j])
    {
        cout<<num<<" " "<"is on location"<<" " "<j<<endl;
    }
}
return 0;
}
```

Output:



```
C:\Users\Masti Malkal\Documents\dd.exe
Enter values:
7
8
9
5
6
Enter a number to find:
6
6 is on location 4
```