DHA Suffa University
Department of Computer Science
CS 2001L – Data Structures and Algorithms Lab
Spring 2021

Lab 01 – Arrays

Objective:

To learn about

- Arrays
- Array operations (Creation, Insertion and Deletion)
- Static Arrays
- Dynamic Arrays

Arrays

An array is a list of finite number of elements stored in the memory. In a linear array, we can store only homogeneous data elements. Elements of the array form a sequence or linear list that can have the same type of data.

Each element of the array is referred by an index. And, the total number of elements in the arraylist is the length of an array. We can access all these elements with the help of the index set.

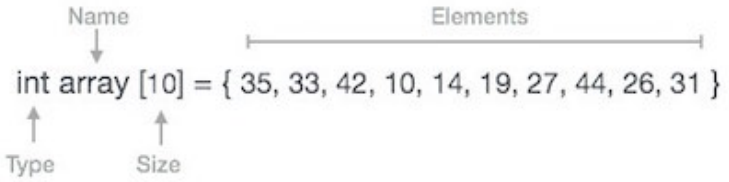

Figure 1.1: Initialization of an array

Declaring Arrays:

Syntax:

Type array_Name [array_Size];

Example:

double balance [10];

Initializing 1-D Arrays:

double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0}; OR

double balance[] = {1000.0, 2.0, 3.4, 17.0, 50.0};

Accessing 1-D Array Elements:

balance[4] = 50.0;

double salary = balance[4];

Static Arrays

Statically declared arrays are allocated memory at compile time and their size is fixed, i.e., cannot be changed later.

Insertion in Array:

At the Beginning:

```
#include <iostream>
#define MAX 5
using namespace std;
int main() {
   int array[MAX] = {2, 3, 4, 5};
   int i = 0;         // loop variable
   int value = 1;     // new data element to be stored in array

   // print array before insertion
   cout<<"Printing array before insertion"<<endl;
   for(i = 0; i < MAX; i++) {
        cout<<"Array["<<i<<"]: "<<array[i]<<endl;
   }

   // now shift rest of the elements downwards
   for(i = MAX-1; i >= 0; i--) {
      array[i+1] = array[i];
   }

   // add new element at first position
   array[0] = value;

   // print to confirm
   cout<<"Printing array after insertion"<<endl;

   for(i = 0; i < MAX; i++) {
      cout<<"Array["<<i<<"]: "<<array[i]<<endl;
   }

return 0;
}
```

At a given index:

```cpp
#include <iostream>
#define MAX 5
using namespace std;
int main() {
    int array[MAX] = {1, 2, 4, 5};
    int i = 0;          // loop variable
    int index = 2;      // index location to insert new value
    int value = 3;      // new data element to be inserted

    // print array before insertion
    cout<<"Printing array before insertion"<<endl;
    for(i = 0; i < MAX; i++) {
         cout<<"Array["<<i<<"]: "<<array[i]<<endl;
    }

    // now shift rest of the elements downwards
    for(i = MAX-1; i >= index; i--) {
       array[i] = array[i-1];
    }

    // add new element at first position
    array[index] = value;

    // print to confirm
    cout<<"Printing array after insertion"<<endl;

    for(i = 0; i < MAX; i++) {
       cout<<"Array["<<i<<"]: "<<array[i]<<endl;
    }
    return 0;
}
```

Deletion from Array:

```
#include <iostream>

using namespace std;

int main() {
    int LA[] = {1,3,5,7,8};
    int j = 3, n = 5;
    int i;

    cout<<"The original array elements are: "<<endl;

    for(i = 0; i<n; i++) {
       cout<<"Array["<<i<<"]: "<<LA[i]<<endl;
    }

    while( j < n) {
       LA[j-1] = LA[j];
       j = j + 1;
    }

    n = n -1;

    cout<<"The array elements after deletion: "<<endl;

    for(i = 0; i<n; i++) {
       cout<<"Array["<<i<<"]: "<<LA[i]<<endl;
    }
    return 0;
}
```

Dynamic Arrays

The dynamic arrays are the arrays which are allocated memory at the runtime. They can have any size.

Syntax:

type *pointerName = new type [arraySize];

Example:

int *ptr = new int[10];

Creating a Dynamic Array:

We need to provide the size of array to the createArray function and it creates the array of provided size and returns the base address of the array.

```
int*  createArray(int size)
{
    int* arr = new int[size];

    for(int i=0;i<size;++i)
        arr[i] = 0;

    return arr;
}
```

Insertion in Array:

```
void insertAt(int* arr, int size,int index,int value)
{
    int i = 0;

    for(i=size-1;i>index;--i)
    {
        arr[i] = arr[i-1];
    }

    arr[index] = value;

}
```

Deletion from Array:

```
void deleteFrom(int* arr,int size,int index)
{
    int i = 0;

    for(i=index;i<size-1;i++)
    {
        arr[i] = arr[i+1];
    }

}
```

Printing an Array:

```
void printArray(int* arr,int size)
{
    for(int i=0;i<size;++i)
    std::cout<<"arr["<<i<<"] = "<<arr[i]<<std::endl;
}
```

Main Function:

```
int main()
{
    // creating Array
    int size = 0, value = 0, index = 0;
    char ch = 0;

    std::cout<<"Enter size of Array : ";
    std::cin>>size;
    int* arr = createArray(size);

    printArray(arr,size);

    //insertion in Array

    do{
        std::cout<<"Enter value you want to insert : ";
        std::cin>>value;
        std::cout<<std::endl;

        std::cout<<"Enter index you want to insert element at : ";
        std::cin>>index;
        std::cout<<std::endl;

        insertAt(arr,size,index,value);

        std::cout<<"Array After insertion"<<std::endl;

        printArray(arr,size);


         std::cout<<"Wish to insert again?? (y/n) : ";
         std::cin>>ch;
     }
     while(ch!='n');

     //delection from Array

     do{
         std::cout<<"Enter index you want to delete element from : ";
         std::cin>>index;
         std::cout<<std::endl;

         deleteFrom(arr,size,index);

         std::cout<<"Array After deletion"<<std::endl;

         printArray(arr,size);

         std::cout<<"Wish to delete again?? (y/n) : ";
         std::cin>>ch;
     }
     while(ch!='n');

     return 0;
 }
```

LAB ASSIGNMENT

1. Create a function that checks the Array overflow condition before insertion.
   [An *overflow* error occurs when one attempts to append the array using an index value greater than the array size]

2. Create a function that checks the Array underflow condition before deletion.
   [An *underflow* error occurs when one attempts to append an empty array]

SUBMISSION GUIDELINES

- Take a screenshot of each task (code and its output), labeled properly.
- Place all the screenshots and the code files (properly labeled) in a single folder labeled with Roll No and Lab No. e.g. **'cs191xxx_Lab01'**.
- Submit the folder at LMS
- **-100%** policies for plagiarism.