# Lab Manual for Data Structures

Lab-2
List Implementation using Arrays

# Table of Contents

# Lab 2: Dynamic List Implementation and Stack ADT (Abstract Data Type)

## 1. Introduction

In this lab, you will learn about the dynamic list implementation using arrays in C++, ADT (Abstract Data Types) and Stack ADT. A list is a collection of items of the same type. The list can be based on numbers (natural or floating-point), text, dates, times, or such values as long as the items are of the same type.An abstract data type (ADT) is a precise model for a certain class of data structures that have similar actions; or for definite data types of one or more programming languages that have similar semantics. Stack is a data structure which allows placement of elements in Last in First out (LIFO) fashion.

### 1.1 Lists and Dynamic Lists

Lists are such data structures which are used to maintain elements of same data type in a linear/sequential way. A list or series is an abstract data type that implements an ordered collection of values, where the same value can occur more than one time. Static list structures allow only inspection and enumeration of the values of its elements. A dynamic list may allow items to be inserted, replaced, or deleted during the list's existence. Size of dynamic lists can be changed during program execution.

### 1.2 Relevant Lecture Readings

a) Revise Lecture No. 5 and 6 available at \\fs\lectures$ in instructor's folder.
b) From books: C++ Data Structures by Nell Dale (Page 196-199) and Data structures using C++ by D. S. Malik (Page 396-399).

## 2. Activity Time boxing

Table 1: Activity Time Boxing

| Task No. | Activity Name | Activity time | Total Time |
|---|---|---|---|
| 5.1 | Design Evaluation | 20mins | 20mins |
| 6.2 | Walk through tasks | 20mins | 20mins |
| 7 | Practice tasks | 15 mins for task 1, 30 mins for task 2 and 25 mins for task 3 | 70mins |
| 9 | Evaluation Task | 60mins for all assigned tasks | 60mins |

## 3. Objectives of the experiment

- To get basic understanding of dynamic lists using arrays in C++
- To write programs related to dynamic lists using arrays in C++ using Microsoft Visual Studio 2017 environment.
- To get an understanding of identifying basic errors in a C++ program by using debugging techniques from program representing dynamic list using arrays.

## 4. Concept Map

This concept map will help students to understand the main concepts of topic covered in lab.

### 4.1 Dynamic Lists in C++ using Arrays

Dynamic list in C++ can be designed using arrays. We may add or remove elements from this list during execution of program and arrays size can controlled at program execution time.

Following code segment represents a list using arrays:

```
const int MaxItems = 20;
class CListOfNumbers
{
private:
     double Item[MaxItems];

public:
     CListOfNumbers();
     ~CListOfNumbers();
};
```

You are required to count the number of elements in list after performing each add or delete operation on list.

```
class CListOfNumbers
{
private:
     double Item[MaxItems];
     int Size;

public:
     int Count() const;
     CListOfNumbers();
};

CListOfNumbers::CListOfNumbers()
{
     Size = 0;
}
intCListOfNumbers::Count() const
{
     return Size;
}
```

Creating a list, consists of adding elements to it. Elements are usually added one at a time and easiest way to do this is to add an item at the end of the list.

To add an item to the list, you first need to test whether the list is already full or not. A list is full if its count of items is equal to or higher than the maximum number you had set. If the list is not empty, you can add an item at the end and increase the count by one. Following code segment represent this functionality.

```
Bool ListOfNumbers::Add(double item)
     {
     if( size < 20 )
          {

               Item[size] = item;
               size++;

               return true;
          }

     return false;
     }
```

After adding elements to a list, you can retrieve them to do what you intended the list for. To retrieve an item, you can locate an item by its position. By using index of this array, you can check if the position specified is negative or higher than the current total of elements. It means index is out of range. If the index is in the right range, you can retrieve its item against that index. Following code segment represents this functionality.

```
Double ListOfNumbers::Retrieve(int pos)
     {
          if(pos>= 0 &&pos<= size )
               return Item[pos];

          return 0;
     }
```

Inserting a new item in the list allows you to add it at a position of you chooses. To insert a new element in the list, you must provide the new item and the desired position. Before performing this operation, you must first check two conditions. First, the list must not be empty. Second, the specified position must be in the valid range. Following code segment provides implementation of this method.

```
Bool ListOfNumbers::Insert(double item, int pos)
     {
     if( size < 20 &&pos>= 0 &&pos<= size )
          {
               for(int i = size; i > pos-1; i--)
                    Item[i+1] = Item[i];
```

```
                Item[pos] = item;

                size++;

                return true;
            }

        return false;
        }
```

Another operation you may perform on a list consists of deleting an element. To delete an item from the list, you need to provide its position in the list. Before performing the operation, you can first check that the specified position is valid. Following code segment provides implementation.

```
Bool ListOfNumbers::Delete(int pos)
        {
        if(pos>= 0 &&pos<= size )
            {

                for(int i = pos; i < size; i++)
                    Item[i] = Item[i+1];

                size--;

                return true;
            }

        return false;
        }
```

## 5. Homework before Lab

This homework will help students to study the concepts of topic before start of lab.

### 5.1 Problem Solution Modeling

After studying the introduction and concept map sections you should be ready to provide the solution of following problems. Design the solutions of the problems in C++ and bring your code to lab so that lab instructor should access and grade it.

### 5.2Problem description:

Design a dynamic list using array whose each element is an object of “person” class. “person” class should have some privately defined attributes: per_id (int),

per_name(string) and per_age (int). Some member functions which are defined publicly are: constructor function which should initialize the attributes of "person" object using blank and zero values, input() function which should allow user to provide input for attributes of object, output() function which should allow user to display the values of attributes of object. This dynamic array should allow insertion, removal, count of total element and displaying the values of elements of list.

### 5.3Practices from home

#### *5.3.1 Task-1*

Make a list of at least 5 benefits we may get while using dynamic lists.

#### *5.3.2 Task-2*

Provide comparison between implementation of lists using static and dynamic arrays.

## 6. Procedure& Tools

This section provides information about tools and programming procedures used for the lab.

### 6.1 Tools

Microsoft Visual Studio 2017 with Visual C++ compiler configured.

### 6.2 Walk through Tasks [Expected time = 20mins]

Dynamic list using array can be implemented in visual studio 2017. Following figure 2 shows screens containing this code.

```
#include <iostream>
#include <conio.h>
#include <iomanip>
using namespace std;


class ListOfNumbers
{
   private: int size;
            double *Item;

   public:
            ListOfNumbers();
            bool Insert(double item, int pos);
};

ListOfNumbers::ListOfNumbers()
{
        cout<<"Enter size of array (from 1 to 20): ";
        cin>>size;
        Item = new double[size];

}
```

**Figure 2: Dynamic list using array in Microsoft Visual Studio 2017.**

Operations related to list such as addition, removal, element count and element retrieval are implemented in this program as well. Figure 3 shows some of these operations.

```
bool ListOfNumbers::Insert(double item, int pos)
{
        if ( pos >= 0 && pos < size )
        {
             for (int i = size; i > pos - 1; i--)
                  Item[i + 1] = Item[i];

             Item[pos] = item;
             return true;
        }

        return false;

}

int main ()
{
      ListOfNumbers ls;
      ls.Insert(2.3, 0);
      ls.Insert(2.5, 1);
      ls.Insert(2.7, 2);
      ls.Insert(3.5, 3);
      ls.Insert(2.2, 4);

      getche();
      return 0;


}
```

## 7. Practice Tasks

This section will provide information about the practice tasks which are required to be performed in lab session. Design solutions of problems discussed in each task and place solution code in a folder specified by your lab instructor.

**Lab instructors are guided to help students learn how ACM problems work and provide students with certain practice/home tasks on the pattern of ACM Problems.**

### 7.1 Practice Task 1 [Expected time = 15mins]

Partial solution of a List Class is given below. Complete the missing code in the functions to run the program properly.

```
int *elements;
int size;
int length;
////////////////////////////////// Constructor
List(int maxsize)
{
      size=maxsize;
      elements=new int[size];
      length=0;
}
```

```
//////////////////////////////////// Destructor
~List ()
{
	delete []elements;
}
////////////////////////////////////  Output the list structure
void showStructure ()
{
	if(!isEmpty())
	{
		for(int i=0;i<length;i++)
		{
			cout<<"Element:"<<elements[i]<<endl;
		}
	}
	else
	{
			cout<<"Display: No items to be displayed. List is empty\n";
	}
}
//////////////////////////////////// List manipulation operations
// Insert after cursor
void insert (int newDataItem)
{
	if(!isFull())
	{
		elements[length]=newDataItem;
		length++;
	}
	else
	{
		cout<<"Insert: Cannot insert more items. List is full\n";
	}
}
// Remove data item
int remove ()
{
	if(!isEmpty())
	{
		length--;
		return elements[length];
	}
	else
	{
		cout<<"Remove: Cannot remove the item. List is empty\n";
	}
}
// Replace data item
void replace ( int newDataItem, int position )
{
	//Condition: List contains at least position+1 data items.
	//Result: Removes the data item present at the position from a list and
replace the number requested by user at its position.
	if(length<position)
```

```
	{
		cout<<"Replace: Cannot replace the item. Invalid position requested\n";
	}
	else
	{
		//implement your logic here
	}
}
// find any number
bool find ( int searchDataItem )
{
	//Condition: List is not empty.
	//Result: Check if the number is present in the list or not

	if(!isEmpty())
	{
		for(int i=0;i<length;i++)
		{
			//implement your logic here
		}
	}

}


////////////////////////////////////// List status operations

// check if List is empty
bool isEmpty ()
{
	// implement your logic here
}

// check if List is full
bool isFull ()
{
	// implement your logic here
}
```

**7.2 Practice Task 2** **[Expected time = 30mins]**

Create a list of at least 10 books (Title, Author, Price, Book_id) with the following functions:

1) Sequential insert in the list on the basis of Book_id
2) Sequential delete in the list and then adjust the list afterwards
3) Insert a value at a specific index and adjust remaining list afterwards
4) Delete a value at a specific index and adjust remaining list afterwards
5) Show all values present in the list
6) Show the length of the list

### 7.3Practice Task 3 [Expected time = 25mins]

Create a list of Employees (List can be as long as user wants). Program should save the following information for each Employee: Name, Emp. No., Experience, Designation and Salary. Now perform the following tasks:

a) Your program should save the information of only those employees whose experience is at least 2 years.
b) Display only those Employees whose Salary is greater than 50,000
c) Display the Employees alphabetically

### 7.4 Out comes

After completing this lab, student will be able to understand the concepts related to dynamic list using arrays and stack abstract data type. They will be able to develop programs related to dynamic lists using arrays in C++ using Microsoft Visual Studio 2017 environment.

### 7.6Testing

**Test Cases for Practice Task-1**

| Sample Input | Sample Output |
|---|---|
| 2 | |
| 3 | |
| 4 | |
| 6 | |
| 9 | |
| Delete an element from location: 3 | Element deleted |
| Enter a value to search in list: 9 | 9 is at location 3 in list |

**Test Cases for Practice Task-2**

| Sample Inputs-1 | Sample Outputs-1 |
|---|---|
| Input elements in list:<br>Bookno : B123<br>Title: C++ programming<br>Price: 234.95<br>Author: Jone Sage | |
| Bookno: B345<br>Title: Object Oriented Programming in C++<br>Price: 235.95<br>Author: B J Shamy | |
| Bookno: B133<br>Title: Data Structures using C and C++<br>Price: 400.35<br>Author: Dietel J. | |

| | |
|---|---|
| Enter an index to place a in list:<br>7<br>Please enter the information of book:<br>Bookno: B133<br>Title: Data Structures using C and C++<br>Price: 400.35<br>Author: Dietel J. | Book with bookno: B133 is Placed at location 7 in list |
| Enter an index to place a book in the list:<br>3 | Sorry the entered index has already a book. |

**Test Cases for Practice Task-3**

| Sample Input | Sample Output |
|---|---|
| Enter the size of list:<br>5<br>Press 1 to add an employee<br>Press 2 to Check the employees having experience more than 2 years<br>Press 3 to display all the employee information | Sorry the list is empty yet. |
| Press 1 to add an employee<br>Press 2 to Check the employees having experience more than 2 years<br>Press 3 to display all the employee information | Please enter the information of employee: |

## 8.Evaluation Task (Unseen) **[Expected time = 60mins for two tasks]**

The lab instructor will give you unseen task depending upon the progress of the class.

## 9. Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

Table 2: Evaluation of the Lab

| Sr. No. | Task No | Description | Marks |
|---|---|---|---|
| 1 | 4 | Problem Modeling | 20 |
| 2 | 6 | Procedures and Tools | 10 |
| 3 | 7,8 | Practice tasks and Testing | 35 |
| 4 | 8.1 | Evaluation Tasks (Unseen) | 20 |
| 5 | | Comments | 5 |
| 6 | | Good Programming Practices | 10 |

## 10. Further Readings

### 10.1 Web sites related to C++ tutorials related to dynamic lists

1. http://www.functionx.com/cpp/Lesson14.htm
3. http://www.element14.com/community/community/code_exchange/blog/2013/03/20/c-tutorial--dynamic-arrays

### 10.2 Web sites containing supporting material

1. http://www.engppt.com/2012/08/data-structures-with-c-ppt-slides.html