Data Structures

BS (CS) _Fall_2024

Lab_05 Manual

Learning Objectives:

1. Array based list implementation

Array Based List Implementation

Arrays:
An array is used to store elements or data items of similar data types at contiguous memory locations and elements can be accessed randomly using indices of an array. Arrays are efficient when we want to store many elements that too of similar data types.

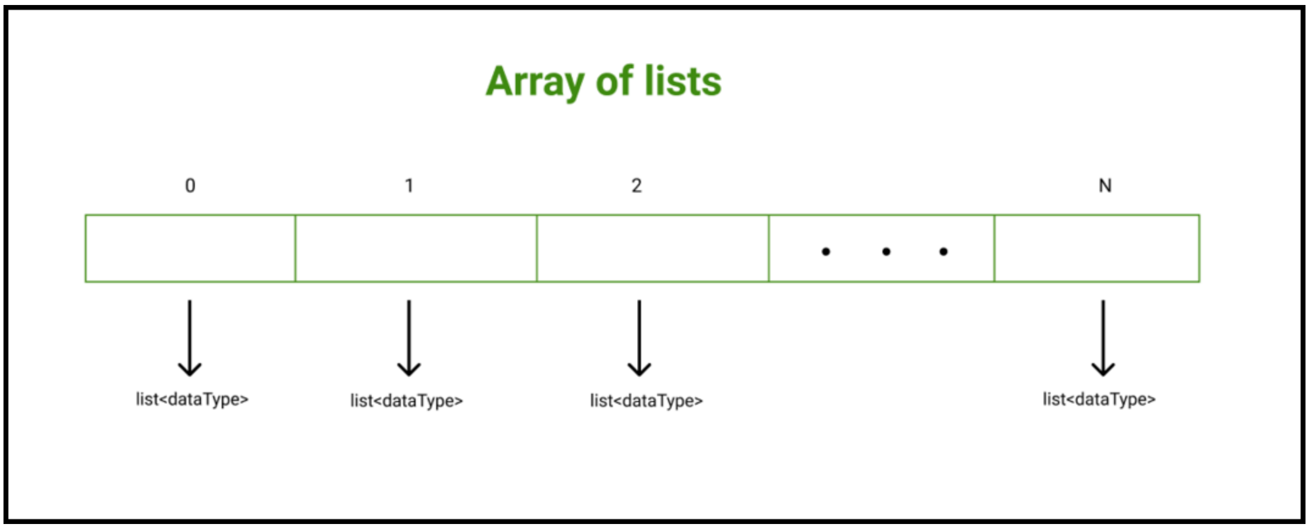
List:
A list is a sequence container that allows non-contiguous memory allocation. If we compare a vector with a list, then a list has slow traversal as compared to a vector but once a position has been found, insertion and deletion are quick. Generally, a list in C++ is a doubly-linked list.

Array of list:
In C++, it allows us to create an array of lists. An array of lists is one in which each element is a list on its own. It is a data structure that stores data in a single array. Also known as backing array.

Syntax

```
list<dataType> Name_of_list[N];

//Here,N: The size of the array of the lists.
//dataType: It signifies that each list can store elements of this data type only.
```



Advantage:
They allow us to randomly access any item in list by providing constant time access to any value in array.

More formally, we can define a list as follows:

List: A collection of elements of the same type.

The length of a list is the number of elements in the list. Following is some of the operations performed on a list:

1. Create the list. The list is initialized to an empty state.
2. Determine whether the list is empty.
3. Determine whether the list is full.
4. Find the size of the list.
5. Destroy, or clear, the list.
6. Determine whether an item is the same as a given list element.
7. Insert an item in the list at the specified location.
8. Remove an item from the list at the specified location.
9. Replace an item at the specified location with another item.
10. Retrieve an item from the list from the specified location.
11. Search the list for a given item.

Because all the elements of a list are of the same type, an effective and convenient way to process a list is to store it in an array. The size of the array can be specified when a list object is declared. It follows that, to maintain and process the list in an array, we need the following three variables:

• The array holding the list of elements
• A variable to store the length of the list (that is, the number of list elements currently in the array)
• A variable to store the size of the array (that is, the maximum number of elements that can be stored in the array)

Array based list implementation using pointers

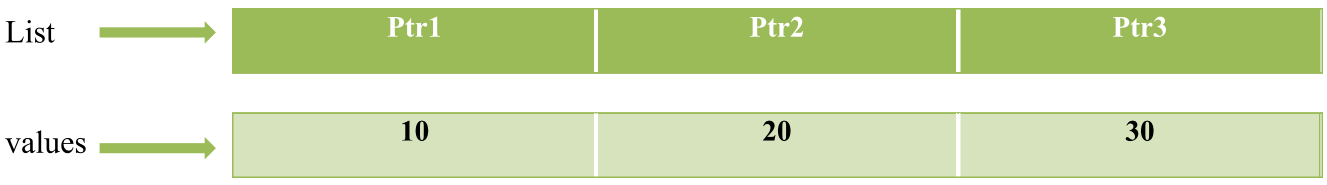
Insert elements in the list

```
#include <iostream>
using namespace std;

class List {
private:
    int **list;      // Pointer to an array of pointers
    int size;        // Maximum size of the list
    int count;
public:
    // Initialize the list
    List(int size) {
        this->size = size;
        count = 0;
        list = new int*[size];  // Allocating memory for an array of integer
pointers
    }

    // Insert new element in the list
    bool insert(int value) {
        if (count >= size) {
            cout << "List is Full" << endl;
            return false;
        }
        list[count] = new int(value);
        count++;
        return true;
    }
int main() {
    List myList(5);

    cout<<"Inserting elements in the list: " << endl;
    myList.insert(10);
    myList.insert(20);
    myList.insert(30);
    return 0;
}
```



Display elements of the list

```
#include <iostream>
using namespace std;

class List {
private:
    int **list;      // Pointer to an array of pointers
    int size;        // Maximum size of the list

public:
    // Initialize the list
    List(int size) {
        this->size = size;
        list = new int*[size];  // Allocating memory for an array of integer
pointers
    }
    #// Display elements of the list
    void display() {
        for (int i = 0; i < size; i++) {
            cout << *list[i] << " ";
        }
        cout << endl;
    }

int main() {
    List myList(5);
    cout<<”The elements in the list are: ” << endl;
    myList.display();

    return 0;
}
```