Data Structures-LAB

Lab Manual # 02

Array Based List in C++

Instructor: Fariba Laiq

Semester Fall-2023

Course Code: CL2001

Fast National University of Computer and Emerging Sciences Peshawar

Department of Computer Science & Software Engineering

Creating an Array-Based List

The provided code defines the ArrayList class, which encapsulates the behavior of an array-based list. It includes methods for managing the list's size, adding, removing, searching, and retrieving elements. In the ArrayList constructor, the size of the list is initialized, and memory is allocated for the array. The curr pointer is also set to NULL. The initial length is set to zero.

- Length indicates the current number of elements in the list.
- Size indicates the total capacity of the list. The number of elements that can be stored in the list.

```
class ArrayList {
                int size;
                int length;
                int *arr;
                int *curr;
        public:
                ArrayList(int size) {
                        this->size=size;
                        length=0;
                        arr=new int[5];
                        curr=NULL;
                }
}
```

Methods with array based list

void start()

The start() method is used to position the curr pointer at the beginning of the array-based list. This method sets the curr pointer to point to the first element of the arr array.

```
void start() {
        curr=arr;
    }
```

void tail()

The tail() method is used to position the curr pointer at the end of the array-based list. It calculates the address of the last element in the list by adding length - 1 to the address of the arr array. The curr pointer is then set to point to this calculated address.

```
void tail() {
        curr=arr+length-1;
    }
```

void back()

The back() method decrements the curr pointer, effectively moving it one position backward in the array-based list. This method is used to navigate to the previous element in the list.

```
void back() {
        curr--;
    }
```

void next()

The next() method increments the curr pointer, moving it one position forward in the array-based list. This method is used to navigate to the next element in the list.

```
void next() {
        curr++;
    }
```

Inserting Elements in Array-Based List

The insert method in the ArrayList class enables the addition of elements at specified positions. When adding an element, several conditions are checked to ensure a valid insertion:

- If the list is full (size equals length), a message "list is full" is displayed.
- If the position is less than 1 or greater than length + 1, indicating an invalid position, "invalid position" is displayed.
- Otherwise, the insertion takes place by shifting elements to accommodate the new value. The tail method is used to position the curr pointer at the last

element. A loop iterates from the end, shifting elements as needed, and the new value is added at the specified position. The length is incremented.

```
void insert(int pos, int val) {
        if(size==length) {
                cout<<"list is full"<<endl;
        } else if(pos<1 || pos>length+1) {
                cout<<"invalid position"<<endl;
        } else {
                tail();
                for(int i=length; i>=pos; i--) {
                        *(curr+1)=*curr;
                        back();
                }
                *(curr+1)=val;
                length++;
        }
}
```

Searching Elements in Array-Based List

The search method searches for a specified value within the array-based list. It iterates through the elements, comparing each with the target value. If a match is found, the position is displayed. This operation helps identify the positions of elements containing a specific value.

```
void search(int val) {
        start();
        for(int i=1; i<=length; i++) {
                if(*curr==val) {
                        cout<<"element found at position: "<<i<<endl;
                        }
                next();
        }
}
```

Retrieving Value at a Specific Position

The get method in the ArrayList class retrieves the value at a specified position. The provided position is validated, and if it's valid, the curr pointer is moved to the desired position. The value at that position is then returned.

```
int get(int pos) {
        if(pos<1 || pos>length) {
                cout<<"invalid position"<<endl;
        } else {
                start();
                for(int i=1; i<pos; i++) {
                        next();
                }
        }
        return *curr;
}
```

Printing the List

As defined in the code, this method begins by invoking the start() method, which positions the curr pointer at the beginning of the list. Using a for loop, we iterate through the list, from the first element to the last. During each iteration, the value pointed to by the curr pointer is printed using the cout statement, followed by a space. After printing the current element, the next() method advances the curr pointer to the next element, thus preparing it for the next iteration. This process continues until all elements within the list have been printed.

```
void printList() {
        start();
        for(int i=1; i<=length; i++) {
                cout<<*curr<<" ";
                next();
        }
        cout<<endl;
}
```

Overall Code:

Note that that below code does not contain the implementation of the remove() method. Please implement it on your own.

```
#include<iostream>
using namespace std;
class ArrayList {
		int size;
		int length;
		int *arr;
		int *curr;
	public:
		ArrayList(int size) {
			this->size=size;
			length=0;
			arr=new int[5];
			curr=NULL;
		}
		void start() {
			curr=arr;
		}
		void tail() {
			curr=arr+length-1;
		}
		void back() {
			curr--;
		}
		void next() {
			curr++;
		}
		void insert(int pos, int val) {
			if(size==length) {
				cout<<"list is full"<<endl;
			} else if(pos<1 || pos>length+1) {
				cout<<"invalid position"<<endl;
			} else {
				tail();
				for(int i=length; i>=pos; i--) {
					*(curr+1)=*curr;
					back();

				}

				*(curr+1)=val;

				length++;

			}

		}
```

```
        void printList() {
                start();
                for(int i=1; i<=length; i++) {
                        cout<<*curr<<" ";
                        next();
                }
                cout<<endl;
        }
        void search(int val) {
                start();

                for(int i=1; i<=length; i++) {

                        if(*curr==val) {
                                cout<<"element found at position: "<<i<<endl;
                        }
                        next();
                }
        }

        int get(int pos) {
                if(pos<1 || pos>length) {
                        cout<<"invalid position"<<endl;
                } else {
                        start();
                        for(int i=1; i<pos; i++) {
                                next();

                        }
                }
                return *curr;
        }
};
```

```
int main() {
        ArrayList l(5);
        l.insert(1, 2);
        l.printList();
        l.insert(2, 3);
        l.printList();
        l.insert(2, 4);
        l.printList();
        l.remove(2);
        l.printList();
        l.search(3);
        cout<<l.get(1);
}
```

Output:

```
2
2 3
2 4 3
2 3
element found at position: 2
2
```