**Department of Computing**

**School of Electrical Engineering and Computer Science**

**CS250 – Data Structures and Algorithms**



**Lab 5: Use Knowledge of Linked Lists to   
Implement a Small Functionality**

**Submission Details**

|  |  |
| --- | --- |
| Name | CMS ID |
| Muhammad Umer | 345834 |
| Group | GP – 1 |
| Lab Engineer | Anum Asif |
| Faculty Member | Bostan Khan |
| Class | BEE12 |
| Date | 04/03/2024 |
| Time | 10:00 am – 12:50 pm |

**Table of Contents**

[2 Use Knowledge of Linked Lists to Implement a Small Functionality 3](#_Toc160440546)

[2.1 Introduction 3](#_Toc160440547)

[2.2 Objectives 3](#_Toc160440548)

[2.3 Tools/Software Requirement 3](#_Toc160440549)

[2.4 Description 3](#_Toc160440550)

[2.5 Deliverables 3](#_Toc160440551)

[3 Lab Task 4](#_Toc160440552)

[4 Conclusion 13](#_Toc160440553)

# Use Knowledge of Linked Lists to Implement a Small Functionality

## Introduction

Doubly circular linked lists are important data structures which are used in many applications. Students have learned the fundamental concepts of circular linked lists in the lectures. This lab will introduce students with the practical implementation of a doubly circular linked list and different operations that can be performed on it.

## Objectives

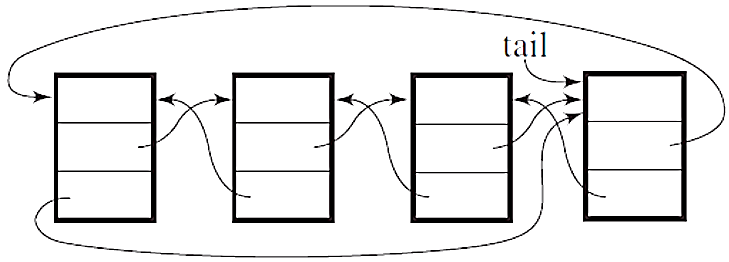
Objective of this lab is to familiarize students with the doubly circular linked list and implement them.

## Tools/Software Requirement

* Visual Studio C++

## Description

This lab is about the implementation of the doubly linked list and circular linked list. In the doubly linked list you have two pointers in the link part of node i.e. next and previous pointers. With the help of these pointers, it becomes very easy to transverse the list. In circular linked list the successor of tail pointer points to the head pointer or start of the linked list. And every new node is added after the last node but before the first node. Keeping these concepts in mind, solve the lab tasks.



The basic operation consists of:

* ***Creating*** the list.
* ***Inserting*** nodes at the beginning, at the end and at a a specific location.
* ***Deletion*** nodes at a specific location.
* ***Traversing/displaying*** the data in the list.

## Deliverables

Compile a single word document by filling in the solution parts and submit this file on LMS. The name of word document should follow this format. i.e., YourFullName(reg)\_Lab#. You must show the implementation of the tasks in a complete manner to get your work graded.

# Lab Task

Use the partially written C++ code to implement a doubly circular linked list. The following are the important features (to be implemented):

1. Create a doubly circular linked list.
2. Create a function which can insert data at the beginning of the list.
3. Create a function which can insert data at the end of the list.
4. Create a function which can insert data at any location within the list.
5. Create a function which can delete a node at a specific location.
6. Create a function that can display the contents of the linked list.

**Partial Code**

/\* C++ Program to Implement Circular Doubly Linked List

\*/

**#include**<iostream>

**#include**<cstdio>

**#include**<cstdlib>

**using** **namespace** std;

/\*

\* Class Declaration

\*/

**class** DoubleCircularList

{

**public**:

**int** counter = 0;

/\*

\* Node Declaration

\*/

**struct** node{

**int** info;

**struct** node \*next;

**struct** node \*prev;

}\*start, \*last;

**DoubleCircularList**(){

start = NULL;

last = NULL;

}

/\*

\*CREATE NODE AND ALLOCATE MEMORY DYNAMICALLY

\*/

node\* **createNode**(**int** value){

**struct** node \*temp;

//your code here

cout << "++++++ You have to write code for createNode()";

**return** temp;

}

/\*

\*INSERTS ELEMENT AT BEGINNING

\*/

**void** **insertAtStart**(){

**int** value;

//cout<<endl<<"Enter the element to be inserted: ";

//cin>>value;

//your code here

cout << "++++++ You have to write code for insertAtStart()";

}

/\*

\*INSERTS ELEMNET AT LAST

\*/

**void** **insertLast**(){

**int** value;

//cout<<endl<<"Enter the element to be inserted: ";

//cin>>value;

//your code here

cout << ++++++ You have to write code for insertLast()";

}

/\*

\*INSERTS ELEMENT AT POSITION

\*/

**void** **insertAtPos**(){

**int** value, pos, i;

cout<<**endl**<<"Enter the element to be inserted: ";

cin>>value;

cout<<**endl**<<"Enter the postion of element inserted: ";

cin>>pos;

//your code here

cout << ++++++ You have to write code for insertAtPos()";

}

/\*

\* Delete Node at Particular Position

\*/

**void** **deleteAtPos**()

{

**int** pos, i;

node \*ptr, \*s;

**if** (start == last && start == NULL)

{

cout<<"List is empty, nothing to delete"<<**endl**;

**return**;

}

cout<<**endl**<<"Enter the postion of element to be deleted: ";

cin>>pos;

//your code here

cout << ++++++ You have to write code for deleteAtPos()";

}

/\*

\* Display Elements of the List

\*/

**void** **display**(){

//your code here

cout << ++++++ You have to write code for display()";

}

};

/\*

\* Main: Contains Menu

\*/

**int** **main**(){

**int** userChoice;

DoubleCircularList cdl;

**while** (1){

cout<<"\n-------------------------------"<<**endl**;

cout<<"Operations on Doubly Circular linked list"<<**endl**;

cout<<"\n-------------------------------"<<**endl**;

cout<<"1.Insert at Beginning"<<**endl**;

cout<<"2.Insert at Last"<<**endl**;

cout<<"3.Insert at Specific Position"<<**endl**;

cout<<"4.Delete at Specific Position"<<**endl**;

cout<<"5.Display List"<<**endl**;

cout<<"6.Exit"<<**endl**;

cout<<"Enter your choice : ";

cin>>userChoice;

**switch**(userChoice){

**case** 1:

cdl.insertAtStart();

**break**;

**case** 2:

cdl.insertLast();

**break**;

**case** 3:

cdl.insertAtPos();

**break**;

**case** 4:

cdl.deleteAtPos();

**break**;

**case** 5:

cdl.display();

**break**;

**case** 6:

**exit**(1);

**default**:

cout<<"Wrong choice"<<**endl**;

}

}

**return** 0;

}

Code

*/\**

*C++ Program to implement Circular Doubly Linked List*

*See unsolved code from the lab manual.*

*\*/*

#include <cstdio>

#include <cstdlib>

#include <iostream>

using *namespace* std;

*// Class declaration*

*class* DoubleCircularList {

*public:*

*int* counter = 0;

*// Node declaration*

*struct* node {

*int* info;

*struct* node *\**next;

*struct* node *\**prev;

    } \*start, \*last;

    DoubleCircularList() {

        start = nullptr;

        last = nullptr;

    }

*// Create node and allocate memory*

    node *\**createNode(*int* *value*) {

*struct* node *\**temp = new node; *// Allocate memory for the node*

        temp->info = *value*;

        temp->next = nullptr;

        temp->prev = nullptr;

        return temp;

    }

*// Insert element at the beginning*

*void* insertAtStart() {

*int* value;

        cout << endl << "Enter the element to be inserted: ";

        cin >> value;

*// your code here*

        node \*temp = createNode(value);

        if (start == last && start == nullptr) {

            temp->next = temp;

            temp->prev = temp;

            start = last = temp;

        } else {

            temp->next = start;

            start->prev = temp;

            start = temp;

            start->prev = last;

            last->next = start;

        }

        counter++;

    }

*// Insert element at last*

*void* insertLast() {

*int* value;

        cout << endl << "Enter the element to be inserted: ";

        cin >> value;

*// your code here*

        node \*temp = createNode(value);

        if (start == last && start == nullptr) {

            temp->next = temp;

            temp->prev = temp;

            start = last = temp;

        } else {

            temp->next = start;

            start->prev = temp;

            last->next = temp;

            temp->prev = last;

            last = temp;

        }

        counter++;

    }

*// Insert element at a specific position*

*void* insertAtPos() {

*int* value, pos, i;

        cout << endl << "Enter the element to be inserted: ";

        cin >> value;

        cout << "Enter the postion of element inserted: ";

        cin >> pos;

*// your code here*

        node \*temp = createNode(value);

        node \*iterator = start;

        for (i = 1; i < pos - 1; i++) {

            iterator = iterator->next;

            if (iterator == start) {

                cout << "There are less than " << pos

<< " elements." << endl;

                return;

            }

        }

        temp->next = iterator->next;

        iterator->next = temp;

        temp->prev = iterator;

        temp->next->prev = temp;

        counter++;

    }

*// Delete Node at Particular Position*

*void* deleteAtPos() {

*int* pos, i;

        node \*ptr, \*s;

        if (start == last && start == NULL) {

            cout << "List is empty, nothing to delete" << endl;

            return;

        }

        cout << endl << "Enter the postion of element to be deleted: ";

        cin >> pos;

*// your code here*

        if (pos == 1) {

            ptr = start;

            start = start->next;

            last->next = start;

            start->prev = last;

            delete ptr;

            counter--;

            return;

        }

        ptr = start;

        for (i = 1; i < pos; i++) {

            ptr = ptr->next;

            if (ptr == start) {

                cout << "There are less than " << pos

<< " elements." << endl;

                return;

            }

        }

        if (ptr->next == start) {

            ptr->prev->next = start;

            start->prev = ptr->prev;

            delete ptr;

            counter--;

            return;

        }

        ptr->next->prev = ptr->prev;

        ptr->prev->next = ptr->next;

        delete ptr;

        counter--;

    }

*// Display elements of the list*

*void* display() {

*// your code here*

        node \*iterator = start;

        if (start == last && start == nullptr) {

            cout << "The List is empty, nothing to display." << endl;

            return;

        }

        cout << "The list is: " << endl;

        do {

            if (iterator->next == start) {

                cout << iterator->info;

                break;

            }

            cout << iterator->info << " <-> ";

            iterator = iterator->next;

        } while (iterator != start);

        cout << endl;

        cout << "The number of elements in the list is: "

<< counter << endl;

    }

};

*// Main function*

*int* main() {

*int* userChoice;

    DoubleCircularList cdl;

    while (1) {

        cout << "\n-------------------------------" << endl;

        cout << "Operations on Doubly Circular linked list" << endl;

        cout << "-------------------------------" << endl;

        cout << "1. Insert at Beginning" << endl;

        cout << "2. Insert at Last" << endl;

        cout << "3. Insert at Specific Position" << endl;

        cout << "4. Delete at Specific Position" << endl;

        cout << "5. Display List" << endl;

        cout << "6. Exit" << endl;

        cout << "Enter your choice : ";

        cin >> userChoice;

        switch (userChoice) {

            case 1:

                cdl.insertAtStart();

                break;

            case 2:

                cdl.insertLast();

                break;

            case 3:

                cdl.insertAtPos();

                break;

            case 4:

                cdl.deleteAtPos();

                break;

            case 5:

                cdl.display();

                break;

            case 6:

                exit(1);

            default:

                cout << "Wrong choice" << endl;

        }

    }

    return 0;

}

Output

root@Zonularity:/home/zonularity/dsa# cd "/home/zonularity/dsa/lab\_5/" && g++ task.cpp -o task && "/home/zonularity/dsa/lab\_5/"task

-------------------------------

Operations on Doubly Circular linked list

-------------------------------

1. Insert at Beginning

2. Insert at Last

3. Insert at Specific Position

4. Delete at Specific Position

5. Display List

6. Exit

Enter your choice : 1

Enter the element to be inserted: 1

-------------------------------

Operations on Doubly Circular linked list

-------------------------------

1. Insert at Beginning

2. Insert at Last

3. Insert at Specific Position

4. Delete at Specific Position

5. Display List

6. Exit

Enter your choice : 2

Enter the element to be inserted: 3

-------------------------------

Operations on Doubly Circular linked list

-------------------------------

1. Insert at Beginning

2. Insert at Last

3. Insert at Specific Position

4. Delete at Specific Position

5. Display List

6. Exit

Enter your choice : 3

Enter the element to be inserted: 2

Enter the postion of element inserted: 1

-------------------------------

Operations on Doubly Circular linked list

-------------------------------

1. Insert at Beginning

2. Insert at Last

3. Insert at Specific Position

4. Delete at Specific Position

5. Display List

6. Exit

Enter your choice : 5

The list is:

1 <-> 2 <-> 3

The number of elements in the list is: 3

-------------------------------

Operations on Doubly Circular linked list

-------------------------------

1. Insert at Beginning

2. Insert at Last

3. Insert at Specific Position

4. Delete at Specific Position

5. Display List

6. Exit

Enter your choice : 4

Enter the postion of element to be deleted: 3

-------------------------------

Operations on Doubly Circular linked list

-------------------------------

1. Insert at Beginning

2. Insert at Last

3. Insert at Specific Position

4. Delete at Specific Position

5. Display List

6. Exit

Enter your choice : 5

The list is:

1 <-> 2

The number of elements in the list is: 2

-------------------------------

Operations on Doubly Circular linked list

-------------------------------

1. Insert at Beginning

2. Insert at Last

3. Insert at Specific Position

4. Delete at Specific Position

5. Display List

6. Exit

Enter your choice : 6

# Conclusion

In this lab, we successfully implemented a circularly doubly linked list in C++. We investigated various operations on the list, encompassing insertion, deletion, and traversal. This hands-on experience reinforced our theoretical understanding of linked list structures and behaviors, while also providing valuable insights into dynamic memory allocation and pointer manipulation within the context of C++ data structures.