Neh Samir Joshi

19BEC1112

28/01/2022

**CSE2003**

**Data Structures and Algorithms**

**[LAB]**

**LAB – 3**

**Singly Linked Lists**

**Aim:** To learn how to implement singly linked lists and their operations.

**Software Required:** Code editor (e.g. VS Code, Dev C++), GCC/G++ compiler

**Task 1:** To create a linear singly linked list and perform create, insert, display, count, and delete operations.

**Code:**

#include <iostream>

using namespace std;

struct Node

{

    int data;

    Node \*next;

};

void Print(struct Node \*n)

{

    while (n != NULL)

    {

        cout << n->data << endl;

        n = n->next;

    }

}

void AddAtFront(struct Node \*\*head, int data)

{

    struct Node \*temp = new Node;

    temp->data = data;

    temp->next = \*head;

    \*head = temp;

}

void Append(struct Node \*\*head, int data){

    struct Node \*temp = new Node;

    struct Node \*last = \*head;

    temp->next = NULL;

    temp->data = data;

    while(last->next != NULL) {

        last = last->next;

    }

    last->next = temp;

}

void AddAtPosition(struct Node \*\*head, int position, int data){ *//0 5 10 15 20*

    struct Node \*temp = new Node;

    struct Node \*n = \*head;

    temp->data = data;

    int i = 1;

    while (i < position-1){

        n = n->next;

        i+=1;

    }

    temp->next = n->next;

    n->next = temp;

}

void DeleteNode(struct Node \*\*list, int value){

    struct Node \*n = \*list;

    struct Node \*temp = new Node;

    while(n->next->data!=value){

        n = n->next;

    }

    struct Node \*prev = n;

    struct Node \*del = n->next;

    struct Node \*after = del->next;

    temp = after;

    free(del);

    n->next = temp;

}

int GetLength(struct Node \*\*list){

    int length = 0;

    struct Node \*n = \*list;

    while (n != NULL){

        n = n->next;

        length += 1;

    }

    return length;

}

int Find(struct Node \*\*list, int value){

    struct Node \*n = \*list;

    int length = GetLength(&n);

    int position = 1;

    while(n->data!=value){

        n = n->next;

        position++;

    }

    if (position==0 || position==length){

        cout << "Element doesn't exist!" << endl;

        return 0;

    }

    else {

        return position;

    }

}

int main()

{

    struct Node \*head = new Node;

    head->data = 0;

    head->next = NULL;

    Print(head);

    AddAtFront(&head, 12);

    cout << "Now adding 12" << endl;

    Print(head);

    AddAtFront(&head, 11);

    cout << "Now adding 11" << endl;

    Print(head);

    Append(&head, 13);

    Append(&head, 14);

    cout << "Now appending 13 and 14" << endl;

    Print(head);

    cout << "Now adding 24 at the fourth position" << endl;

    AddAtPosition(&head, 4, 24);

    Print(head);

    cout << "Length of the linked list: " << GetLength(&head) << endl;

    DeleteNode(&head, 24);

    cout << "Deleting 24 from the LL" << endl;

    Print(head);

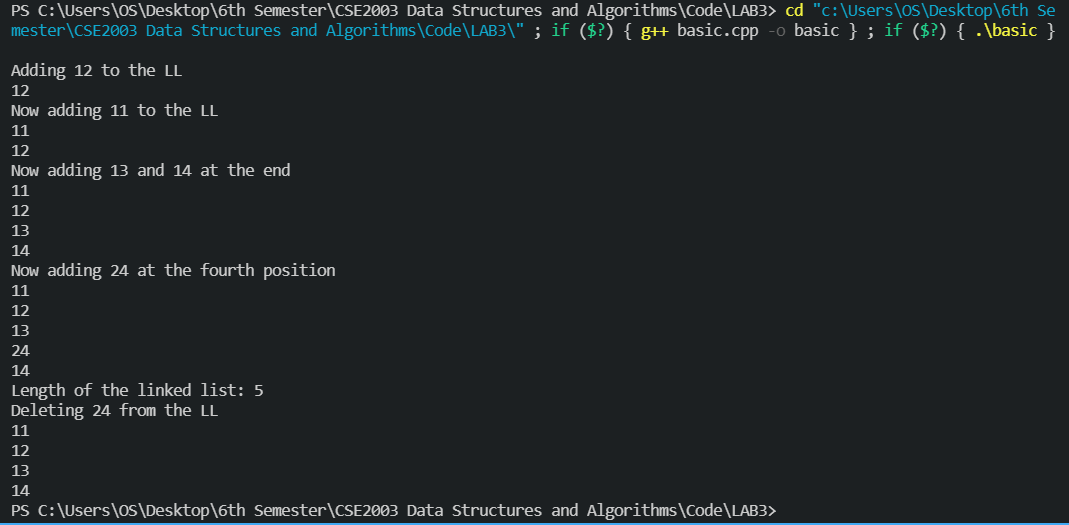
    int position = Find(&head, 27);

    cout << "Position of 13 in the LL is: " << position << endl;

    return 0;

}

**Output:**



**Task 2**: **Implement singly circular linked list and perform create, insert, display, count, and delete operations.**

**Code:**

#include <iostream>

using namespace std;

struct Node

{

    int data;

    Node \*next;

};

void Display(struct Node \*head)

{

    if (head == NULL)

    {

        cout << "Circular linked list is empty!" << endl;

        return;

    }

    struct Node \*initial = head;

    do

    {

        cout << head->data << endl;

        head = head->next;

    } while (head != initial);

}

int GetLength(struct Node \*head)

{

    int length = 0;

    if (head == NULL)

    {

        return length;

    }

    else if (head != NULL && head->next == NULL)

        return 1;

    else

    {

        struct Node \*initial = head;

        head = head->next;

        length = 1;

        while (head != initial)

        {

            length++;

            head = head->next;

        }

        return length;

    }

}

struct Node \*Append(struct Node \*head, int data) *//Will add an element just before the head (and just after the element that points to the head)*

{

    struct Node \*temp = new Node;

    temp->data = data;

    if (GetLength(head) == 0)

    {

        head = temp;

        head->next = head;

    }

    else

    {

        struct Node \*initial = head;

        while (head->next != initial)

        {

            head = head->next;

        }

        temp->next = initial;

        head->next = temp;

        head = initial;

    }

    return head;

}

struct Node \*Front(struct Node \*head, int data)

{

    struct Node \*temp = new Node;

    temp->data = data;

    if (GetLength(head) == 0)

    {

        head = temp;

        head->next = head;

    }

    else

    {

        struct Node \*initial = head;

        while (head->next != initial)

        {

            head = head->next;

        }

        temp->next = initial;

        head->next = temp;

        head = temp;

    }

    return head;

}

struct Node \*Delete(struct Node \*head, int data)

{

    struct Node \*last = head;

    struct Node \*initial = head;

    if (head == NULL)

    {

        cout << "LL is empty!" << endl;

        return head;

    }

    if (head->data == data)

    {

        while (last->next != head)

        {

            last = last->next;

        }

        last->next = head->next;

        head = NULL;

        head = last->next;

        return head;

    }

    while (last->next != head && last->next->data != data)

    {

        last = last->next;

    }

    if (last->next->data == data)

    {

        struct Node \*temp = last->next;

        last->next = temp->next;

        while (last != initial)

            last = last->next;

        return last;

    }

    return head;

}

bool Find(struct Node \*head, int data)

{

    struct Node \*initial = head;

    while (head->next != initial)

    {

        if (head->data == data)

        {

            return true;

        }

        head = head->next;

    }

    return false;

}

struct Node \*InsertAfterPosition(struct Node \*head, int data, int pos)

{

    struct Node \*initial = head;

    int i = 1;

    while (i != pos)

    {

        head = head->next;

        i+=1;

    }

    struct Node \*temp = new Node;

    temp->data = data;

    temp->next = head->next;

    head->next = temp;

    head = initial;

    return head;

}

int main()

{

    struct Node \*head = new Node;

    head = NULL;

    cout << "Adding 10, 20 and 30 to the LL" << endl;

    head = Append(head, 10);

    head = Append(head, 20);

    head = Append(head, 30);

    Display(head);

    cout << "Adding 4 and 5 to the front of the LL" << endl;

    head = Front(head, 5);

    head = Front(head, 4);

    Display(head);

    cout << "Deleting 20 from the LL" << endl;

    head = Delete(head, 20);

    Display(head);

    cout << "Now deleting 4" << endl;

    head = Delete(head, 4);

    Display(head);

    cout << "Now finding 10 in the linked list, searching..." << endl;

    int find = 10;

    if (Find(head, find) == true)

    {

        cout << find << " is in the LL" << endl;

    }

    else

    {

        cout << find << " is not in the LL" << endl;

    }

    cout << "Inserting 212 after 2nd position" << endl;

    head = InsertAfterPosition(head, 212, 2);

    Display(head);

    return 0;

}

**Output:**



**Conclusion**

Thus, we have successfully implemented singly-linear and singly-circular linked lists and perform create, insert, search, count and delete operations of the same. Hence, the experiment is complete.

*Neh Samir Joshi*

*19BEC1112*