A picture containing text

Description automatically generated



Lab Journal: 4

Date: 20 / 10 / 24

Student : Abdul Rafay

Enrollment : 01-131232-004

Department of Software Engineering

Bahria University, Islamabad

Data Structures & Algorithms Lab (Spring-2024)

Teacher: RAHEELA AMBRIN

**Comments:**

**Signature**

### Code:

All the code files are uploaded on GitHub: <https://github.com/CharlieFour/DSA_Lab>

You can check out the code on GitHub in Lab\_05 folder.

### Class file (used in both tasks):

Both tasks are solved in the same way, so I just create only one exe.

#### List.h

typedef struct Node\* Nodeptr;  
  
struct Node{  
    int data;  
    Nodeptr next;  
};  
  
class LinkedList{  
    private:  
        Nodeptr head;  
    public:  
        int length;  
        LinkedList();  
        void traverse();  
        Nodeptr find(int);  
        void iAS(int x);  
        int dAS();  
        void iAE(int x);  
        int dAE();  
        int iAM(int x, int index);  
        int dAM(int index);  
        void saveList();  
        void loadList();  
};

#### List.cpp

#include "list.h"  
#include <iostream>  
#include <fstream>  
  
LinkedList::LinkedList(){  
    head = nullptr;  
    length = 0;  
  
}  
  
void LinkedList::iAS(int x){  
    Nodeptr p = new Node;  
    p->data = x;  
    p->next = head;  
    head = p;  
  
    length++;  
}  
  
int LinkedList::dAS(){  
    if(head == nullptr){  
        return -1;  
    }  
    if(head->next == nullptr){  
        int x = head->data;  
        head = nullptr;  
        return x;  
    }  
    Nodeptr p = head->next;  
    int x = head->data;  
    delete head;  
    head = p;  
  
    length--;  
  
    return x;  
}  
  
void LinkedList::traverse(){  
    for(Nodeptr p = head; p != nullptr; p = p->next){  
        std::clog << p->data << std::endl;  
    }  
}

Nodeptr LinkedList::find(int x){  
    for(Nodeptr p = head; p != nullptr; p = p->next){  
        if(p->data == x){  
            return p;  
        }  
    }  
    return nullptr;  
}  
  
void LinkedList::iAE(int x){  
    if(head == nullptr){  
        iAS(x);  
    }  
    Nodeptr p = head;  
    while(p->next != nullptr){  
        p = p->next;  
    }  
    Nodeptr q = new Node;  
    q->data = x;  
    q->next = nullptr;  
    p->next = q;  
  
    length++;  
}  
  
int LinkedList::dAE(){  
    if(head == nullptr){  
        return -1;  
    }  
    if(length == 1){  
        return dAS();  
    }  
    Nodeptr p = head;  
    Nodeptr q = nullptr;  
    while(p->next != nullptr){  
        if(p->next->next == nullptr){  
            q = p;  
        }  
        p = p->next;  
    }  
    int x = p->data;  
    delete p;  
    q->next = nullptr;  
      
    length--;  
  
    return x;  
}

int LinkedList::iAM(int x, int index){  
    Nodeptr p = head;  
    int i = 0;  
    Nodeptr q = nullptr;  
    while(p->next != nullptr){  
        if(i == index){  
            q = p;  
            break;  
        }  
        i++;  
        p = p->next;  
    }  
  
    if(q == nullptr){  
        return -1;  
    }  
    Nodeptr r = new Node;  
    r->data = x;  
    r->next = q->next;  
    q->next = r;  
  
    length++;  
    return 0;  
}  
  
int LinkedList::dAM(int index){  
    Nodeptr p = head;  
    int i = 0;  
    Nodeptr q = nullptr;  
    while(p->next != nullptr){  
        if(i == index - 1){  
            q = p;  
        }  
        if(i == index){  
            break;  
        }  
        i++;  
        p = p->next;  
    }  
  
    if(q == nullptr){  
        return -1;  
    }  
    int x = p->data;  
    q->next = p->next;  
    delete p;  
      
    length--;  
  
    return x;  
}

void LinkedList::saveList()  
{  
    std::ofstream file("data/list.txt");  
    for(Nodeptr p = head; p != nullptr; p = p->next)  
    {  
        file << p->data << std::endl;  
    }  
    file.close();  
}  
  
void LinkedList::loadList()  
{  
    std::ifstream file("data/list.txt");  
    if(file.is\_open())  
    {  
        int x;  
        while(file >> x)  
        {  
            iAE(x);  
        }  
        file.close();  
    }  
    else  
    {  
        std::cerr << "File not found" << std::endl;  
        system("pause");  
    }  
}

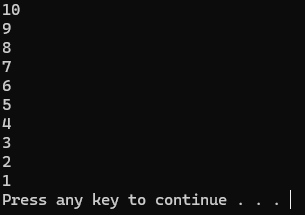
#### Main.cpp

#include <iostream>  
#include <limits>  
#include "../libraries/list.h"  
  
using namespace std;  
  
void takeInput(int &input);  
void printMenu();  
void useList(LinkedList &list);  
  
main()  
{  
    LinkedList list;  
    list.loadList();  
    useList(list);  
    list.saveList();  
    system("pause");  
    return 0;  
}

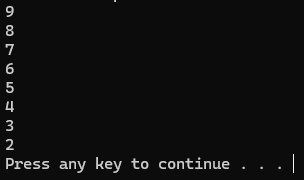
void printMenu()  
{  
    cout << "1. Insertion at the start" << endl;  
    cout << "2. Insertion at the end" << endl;  
    cout << "3. Insertion at the middle" << endl;  
    cout << "4. Deletion from the start" << endl;  
    cout << "5. Deletion from the end" << endl;  
    cout << "6. Deletion from the middle" << endl;  
    cout << "7. Search for an element" << endl;  
    cout << "8. Display the list" << endl;  
    cout << "9. Exit" << endl;  
}  
  
void takeInput(int &input)  
{  
    bool check = false;  
    do  
    {  
        system("cls");  
        printMenu();  
        cout << "Enter the input: ";  
        if(cin >> input)  
        {  
            check = false;  
        }  
        else  
        {  
            cin.clear();  
            cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
            cerr << "Invalid input. Please enter a valid integer." << endl;  
            check = true;  
            system("pause");  
        }  
    }  
    while(check);  
}  
void useList(LinkedList &list)  
{  
    int choice, input, index; // inputList used to input in list   
    do  
    {  
        takeInput(choice);  
        if (choice == 1)  
        {  
            cout << "Enter the element to be inserted: ";  
            cin >> input;  
            list.iAS(input);  
        }

        else if (choice == 2)  
        {  
            cout << "Enter the element to be inserted: ";  
            cin >> input;  
            list.iAE(input);  
        }  
        else if (choice == 3)  
        {  
            cout << "Enter the element to be inserted: ";  
            cin >> input;  
            cout << "Enter the index of the element: ";  
            cin >> index;  
            list.iAM(input,index);  
        }  
        else if (choice == 4)  
        {  
            list.dAS();  
        }  
        else if (choice == 5)  
        {  
            list.dAE();  
        }  
        else if (choice == 6)  
        {  
            cout << "Enter the index of the element: ";  
            cin >> index;  
            list.dAM(index);  
        }  
        else if (choice == 7)  
        {  
            cout << "Enter the element to search for: ";  
            cin >> input;  
            Nodeptr node = list.find(input);  
            cout << ((node == nullptr) ? "Element not found" : "Element found") << endl;   
            system("pause");  
        }  
        else if (choice == 8)  
        {  
            list.traverse();  
            system("pause");  
        }  
    }  
    while(choice != 9);  
}

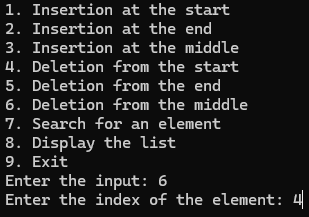
### Screen Shots:

Numbers inserted using insertion function.  


After deleting from start and end.



After deleting from middle.



A black screen with white text

Description automatically generated