A picture containing text

Description automatically generated



Lab Journal: 6

Date: 27 / 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\_06 folder.

#### Linked List .h file:

typedef struct Node\* Nodeptr;  
  
struct Node{  
    int data;  
    Nodeptr next;  
    Nodeptr prev;  
};  
  
class DoubleLinkedList  
{  
    private:  
        Nodeptr list;  
    public:  
        int length;  
        DoubleLinkedList();  
        void traverseBackward();  
        void traverseForward();  
        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();  
        void clear();  
};

#### Linked List CPP file:

#include "doublelinkedlist.h"  
#include <iostream>  
#include <fstream>  
  
DoubleLinkedList::DoubleLinkedList()  
{  
    list = nullptr;  
    length = 0;  
}  
  
void DoubleLinkedList::iAS(int x)  
{  
    Nodeptr p = new Node;  
    p->data = x;  
    p->prev = list;  
    p->next = nullptr;  
    list = p;  
  
    length++;  
}  
  
int DoubleLinkedList::dAS()  
{  
    if(list == nullptr)  
    {  
        return -1;  
    }  
    if(length == 1)  
    {  
        int x = list->data;  
        list = nullptr;  
        return x;  
    }  
    Nodeptr p = list->prev;  
    int x = list->data;  
    delete list;  
    list = p;  
  
    length--;  
  
    return x;  
}

void DoubleLinkedList::traverseBackward()  
{  
    for(Nodeptr p = list; p != nullptr; p = p->prev)  
    {  
        std::clog << p->data << std::endl;  
    }  
}

void DoubleLinkedList::traverseForward()  
{  
    Nodeptr lastNode = list;  
    while (lastNode != nullptr && lastNode->prev != nullptr)  
    {  
        lastNode = lastNode->prev;    
    }  
  
    for (Nodeptr p = lastNode; p != nullptr; p = p->next)  
    {  
        std::clog << p->data << std::endl;  
    }  
}  
  
Nodeptr DoubleLinkedList::find(int x)  
{  
    for(Nodeptr p = list; p != nullptr; p = p->prev)  
    {  
        if(p->data == x)  
        {  
            return p;  
        }  
    }  
    return nullptr;  
}  
  
void DoubleLinkedList::iAE(int x)  
{  
    if(list == nullptr)  
    {  
        iAS(x);  
        return;  
    }  
    Nodeptr p = list;  
    while(p->prev != nullptr)  
    {  
        p = p->prev;  
    }  
    Nodeptr q = new Node;  
    q->data = x;  
    q->prev = nullptr;  
    q->next = p;  
    p->prev = q;  
  
    length++;  
}

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

int DoubleLinkedList::dAM(int index)  
{  
    Nodeptr p = list;  
    int i = 0;  
    Nodeptr q = nullptr;  
    while(p->prev != nullptr)  
    {  
        if(i == index)  
        {  
            q = p;  
        }  
        i++;  
        p = p->prev;  
    }  
  
    if(q == nullptr)  
    {  
        return -1;  
    }  
  
    int x = q->data;  
    q->next->prev = q->prev;  
    q->prev->next = q->next;  
    delete q;  
      
    length--;  
  
    return x;  
}  
  
void DoubleLinkedList::saveList()  
{  
    std::ofstream file("data/list.txt");  
    for(Nodeptr p = list; p != nullptr; p = p->prev)  
    {  
        file << p->data << std::endl;  
    }  
    file.close();  
}  
  
void DoubleLinkedList::loadList()  
{  
    std::ifstream file("data/list.txt");  
    int x;  
    while(file >> x)  
    {  
        iAE(x);  
    }  
    file.close();  
}

void DoubleLinkedList::clear()  
{  
    Nodeptr p = list->prev;  
    while(p->prev != nullptr)  
    {  
        Nodeptr q = p->next;  
        delete p;  
        p = q;  
    }  
}

#### Main:

#include <iostream>  
#include <Windows.h>  
#undef max  
#include <limits>  
#include "../libraries/doublelinkedlist.h"  
  
using namespace std;  
  
void takeInput(int &input);  
void printMenu();  
void useList(DoubleLinkedList &list);  
  
main()  
{  
    SetConsoleTitleA("Double Linked List");  
    DoubleLinkedList 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 forward" << endl;  
    cout << "9. Display the list backward" << endl;  
    cout << "10. 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(DoubleLinkedList &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.traverseForward();  
            system("pause");  
        }  
        else if (choice == 9)  
        {  
            list.traverseBackward();  
            system("pause");  
        }  
    }  
    while(choice != 10);  
}