DSA Lab

Mr. ALEEM AHMAD

A logo of a university

Description automatically generated

Bahria University

**Lab # 6**

**Linked List Implementation**

LAB Journal

Asim Ali (01-131232-015)

**Lab 6:Linked List** **Implementation**

**TASK:**

Linked List Implementation

**Lab Task GitHub Link:**

[Link](https://github.com/iasimkhan2005/DSA.git)

**OUTPUT:**

**MENU**

A screen shot of a computer program

Description automatically generated

**INSERTED AT START(1) AND AT END(5)**

A black background with white text

Description automatically generated

**INSERT AT MIDDLE**

A black screen with white text

Description automatically generated

**AFTER ANY VALUE**

A black background with white text

Description automatically generated  
A black background with white text

Description automatically generated

**BY GIVING POSTION**

A black screen with white text

Description automatically generated

A black screen with white text

Description automatically generated

**CODE:**

#include<iostream>

#include<string>

using namespace std;

int Insertinput() {

system("cls");

int value;

cout << "Enter the value to insert: ";

cin >> value;

if (cin.fail()) {

cin.clear();

cin.ignore();

cout << "Please Enter the Integer Value!!!" << endl;

system("pause");

value = Insertinput();

}

return value;

}

int deleteinput() {

system("cls");

int value;

cout << "Enter the value to Delete: ";

cin >> value;

if (cin.fail()) {

cin.clear();

cin.ignore();

cout << "Please Enter the Integer Value!!!" << endl;

system("pause");

value = deleteinput();

}

return value;

}

class LinkedList {

private:

struct Node

{

int info;

Node\* next;

};

typedef struct Node\* NODEPTR;

NODEPTR listptr ,head;

public:

LinkedList() {

listptr = NULL;

head = NULL;

}

Node\* getnode() {

Node\* newNode = new Node();

}

//insert at Start Of list

void insertAtStart(int value) {

NODEPTR p = nullptr;

p = new Node();

p->info = value;

p->next = head;

head = p;

if (listptr == NULL)

listptr = head;

cout << "Inserted!!" << endl;

}

void deletionFromStart() {

if (head == NULL) {

cout << "List Is EMPTY!!!!!" << endl;

return;

}

else {

NODEPTR temp;

temp = new Node();

temp = head;

head = head->next;

delete temp;

cout << "Deleted!!" << endl;

}

}

//Insert at Middle of list

void insertAtMiddleA(int oldValue, int newValue) {

NODEPTR p,q;

p = new Node();

q = new Node();

for (p = head; p != 0 && p->info != oldValue; p = p->next)

;

if (p == 0) {

cout << "List is empty" << endl;

exit(1);

}

q->info = newValue;

q->next = p->next;

p->next = q;

cout << "Inserted At Middle!!" << endl;

}

void insertAtMiddleB(int postion, int NewValue) {

NODEPTR p = head , q, r;

q = new Node();

r = new Node();

if (p == NULL) {

cout << "List is EMPTY!!" << endl;

return;

}

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

if (i == postion - 1) {

r = p;

}

p = p->next;

}

q->info = NewValue;

q->next = p;

if(q->next == p)

r->next = q;

cout << "Inserted At Middle!!" << endl;

}

void deleteFromMiddle(int value) {

NODEPTR p ,q;

p = new Node();

q = new Node();

for (p = head; p != 0 && p->info != value; p = p->next) {

q = p;

}

if (p == 0) {

cout << "List is Empty" << endl;

return;

}

if (p->info == value) {

q->next = p->next;

delete p;

cout << "deleted!!" << endl;

}

}

//push at end

void push(int value)

{

NODEPTR p = new Node();

p->info = value;

if (head == nullptr) {

head = p;

return;

}

NODEPTR q = head;

while (q->next != nullptr) {

q = q->next;

}

q->next = p;

}

void deleteAtEnd() {

if (head == nullptr) {

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

return;

}

if (head->next == nullptr) {

delete head;

head = nullptr;

return;

}

NODEPTR temp = head;

while (temp->next->next != nullptr) {

temp = temp->next;

}

delete temp->next;

temp->next = nullptr;

cout << "Deleted!!";

}

//Display List

void display()

{

NODEPTR ptr;

ptr = head;

cout << "LIST" << endl;

while (ptr != NULL)

{

cout << ptr->info << "\t";

ptr = ptr->next;

}

}

bool contains(int value) {

NODEPTR temp = head;

while (temp != nullptr) {

if (temp->info == value) {

return true;

}

temp = temp->next;

}

return false;

}

int size() {

int count = 0;

NODEPTR temp = head;

while (temp != nullptr) {

count++;

temp = temp->next;

}

return count;

}

};

int main() {

LinkedList List;

do

{

system("cls");

char option;

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

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

cout << "2.Delete From Start." << endl;

cout << "3.Insert at Middle." << endl;

cout << "4.Delete From Middle." << endl;

cout << "5.Insert at End." << endl;

cout << "6.Delete From End." << endl;

cout << "7.Display List." << endl;

cout << "0.Exit." << endl;

cout << "Option: ";

cin >> option;

if (option == '1')

{

int value;

system("cls");

value = Insertinput();

List.insertAtStart(value);

cout << "Inserted" << endl;

system("pause");

}

else if (option == '2')

{

system("cls");

List.deletionFromStart();

cout << "Deleted" << endl;

system("pause");

}

else if (option == '3')

{

char op;

system("cls");

do

{

cout << "Select the option." << endl;

cout << "1.Inserting after any Value." << endl;

cout << "2.Inserting by Giving Position." << endl;

cout << "Option: ";

cin >> op;

if (op == '1') {

do

{

system("cls");

int OLDvalue, NEWvalue;

cout << "Enter the Old Value: ";

cin >> OLDvalue;

//checking that user input the integer or not!

if (cin.fail()) {

cin.clear();

cin.ignore();

cout << "Please Enter the Integer Value!!!" << endl;

system("pause");

continue;

}

if (List.contains(OLDvalue)) {

}

else {

cout << "Value not found, please enter correct value!" << endl;

continue;

}

cout <<"Enter the New Value: ";

cin >>NEWvalue;

//checking that user input the integer or not!

if (cin.fail()) {

cin.clear();

cin.ignore();

cout << "Please Enter the Integer Value!!!" << endl;

cout << "Enter the New Value: ";

cin >> NEWvalue;

}

List.insertAtMiddleA(OLDvalue,NEWvalue);

break;

} while (true);

break;

}

else if (op == '2')

{

do

{

int pos, NEWvalue;

cout << "Enter the Position: ";

cin >> pos;

if (pos <= List.size()) {}

else {

cout << "Position is Greater than the list size." << endl;

continue;

}

//checking that user input the integer or not!

if (cin.fail()) {

cin.clear();

cin.ignore();

cout << "Please Enter the Integer Value!!!" << endl;

continue;

}

cout << "Enter the Value: ";

cin >> NEWvalue;

//checking that user input the integer or not!

if (cin.fail()) {

cin.clear();

cin.ignore();

cout << "Please Enter the Integer Value!!!" << endl;

continue;

}

List.insertAtMiddleB(pos, NEWvalue);

break;

} while (true);

break;

}

else

{

cout << "Invalid Input!" << endl;

system("pause");

}

} while (true);

}

else if (option == '4')

{

system("cls");

int value;

value = deleteinput();

List.deleteFromMiddle(value);

system("pause");

}

else if (option == '5')

{

system("cls");

int value;

value = Insertinput();

List.push(value);

cout << "Inserted!!";

system("pause");

}

else if (option == '6')

{

system("cls");

cout << "Deleted!!" << endl;

List.deleteAtEnd();

system("pause");

}

else if (option == '7')

{

system("cls");

List.display();

cout << endl;

system("pause");

}

else if (option == '0')

{

exit(1);

}

else {

cout << "Invalid Option,Please Enter Correct one!!!" << endl;

system("pause");

}

} while (true);

}