Program to create a linked list

#include <iostream>

using namespace std;

//node structure

struct Node {

int data;

Node\* next;

Node\* prev;

};

class LinkedList {

public:

Node\* head;

public:

//constructor to create an empty LinkedList

LinkedList(){

head = NULL;

}

};

// test the code

int main() {

//create an empty LinkedList

LinkedList MyList;

//Add first node.

Node\* first = new Node();

first->data = 10;

first->next = NULL;

first->prev = NULL;

//linking with head node

MyList.head = first;

//Add second node.

Node\* second = new Node();

second->data = 20;

second->next = NULL;

//linking with first node

second->prev = first;

first->next = second;

//Add third node.

Node\* third = new Node();

third->data = 30;

third->next = NULL;

//linking with second node

third->prev = second;

second->next = third;

cout << first->data<<endl;

cout <<second->data<<endl;

cout <<third->data<<endl;

return 0;

}

TRAVERSE A DOUBLY LINKED LIST

#include <iostream>

using namespace std;

//node structure

struct Node {

int data;

Node\* next;

Node\* prev;

};

class LinkedList {

public:

Node\* head;

public:

//constructor to create an empty LinkedList

LinkedList(){

head = NULL;

}

//display the content of the list

void PrintList() {

Node\* temp = head;

if(temp != NULL) {

cout<<"The list contains: ";

while(temp != NULL) {

cout<<temp->data<<" ";

temp = temp->next;

}

cout<<endl;

} else {

cout<<"The list is empty.\n";

}

}

};

// test the code

int main() {

//create an empty LinkedList

LinkedList MyList;

//Add first node.

Node\* first = new Node();

first->data = 10;

first->next = NULL;

first->prev = NULL;

//linking with head node

MyList.head = first;

//Add second node.

Node\* second = new Node();

second->data = 20;

second->next = NULL;

//linking with first node

second->prev = first;

first->next = second;

//Add third node.

Node\* third = new Node();

third->data = 30;

third->next = NULL;

//linking with second node

third->prev = second;

second->next = third;

//print the content of list

MyList.PrintList();

return 0;

}

INSERT AN ELEMENT

#include <iostream>

using namespace std;

//node structure

struct Node {

int data;

Node\* next;

Node\* prev;

};

class LinkedList {

private:

Node\* head;

public:

LinkedList(){

head = NULL;

}

//Add new element at the end of the list

void push\_back(int newElement) {

Node\* newNode = new Node();

newNode->data = newElement;

newNode->next = NULL;

newNode->prev = NULL;

if(head == NULL) {

head = newNode;

} else {

Node\* temp = head;

while(temp->next != NULL)

temp = temp->next;

temp->next = newNode;

newNode->prev = temp;

}

}

//Inserts a new element at the given position

void push\_at(int newElement, int position) {

Node\* newNode = new Node();

newNode->data = newElement;

newNode->next = NULL;

newNode->prev = NULL;

if(position < 1) {

cout<<"\nposition should be >= 1.";

} else if (position == 1) {

newNode->next = head;

head->prev = newNode;

head = newNode;

} else {

Node\* temp = head;

for(int i = 1; i < position-1; i++) {

if(temp != NULL) {

temp = temp->next;

}

}

if(temp != NULL) {

newNode->next = temp->next;

newNode->prev = temp;

temp->next = newNode;

if(newNode->next != NULL)

newNode->next->prev = newNode;

} else {

cout<<"\nThe previous node is null.";

}

}

}

//display the content of the list

void PrintList() {

Node\* temp = head;

if(temp != NULL) {

cout<<"The list contains: ";

while(temp != NULL) {

cout<<temp->data<<" ";

temp = temp->next;

}

cout<<endl;

} else {

cout<<"The list is empty.\n";

}

}

};

// test the code

int main() {

LinkedList MyList;

//Add three elements in the list.

MyList.push\_back(10);

MyList.push\_back(20);

MyList.push\_back(30);

MyList.PrintList();

//Insert an element at position 2

MyList.push\_at(100, 2);

MyList.PrintList();

//Insert an element at position 1

MyList.push\_at(200, 1);

MyList.PrintList();

return 0;

}

DELETE A NODE AT A SSPECIFIED POSITION

#include <iostream>

using namespace std;

//node structure

struct Node {

int data;

Node\* next;

Node\* prev;

};

class LinkedList {

private:

Node\* head;

public:

LinkedList(){

head = NULL;

}

//Add new element at the end of the list

void push\_back(int newElement) {

Node\* newNode = new Node();

newNode->data = newElement;

newNode->next = NULL;

newNode->prev = NULL;

if(head == NULL) {

head = newNode;

} else {

Node\* temp = head;

while(temp->next != NULL)

temp = temp->next;

temp->next = newNode;

newNode->prev = temp;

}

}

//Delete an element at the given position

void pop\_at(int position) {

if(position < 1) {

cout<<"\nposition should be >= 1.";

} else if (position == 1 && head != NULL) {

Node\* nodeToDelete = head;

head = head->next;

free(nodeToDelete);

if(head != NULL)

head->prev = NULL;

} else {

Node\* temp = head;

for(int i = 1; i < position-1; i++) {

if(temp != NULL) {

temp = temp->next;

}

}

if(temp != NULL && temp->next != NULL) {

Node\* nodeToDelete = temp->next;

temp->next = temp->next->next;

if(temp->next->next != NULL)

temp->next->next->prev = temp->next;

free(nodeToDelete);

} else {

cout<<"\nThe node is already null.";

}

}

}

//display the content of the list

void PrintList() {

Node\* temp = head;

if(temp != NULL) {

cout<<"The list contains: ";

while(temp != NULL) {

cout<<temp->data<<" ";

temp = temp->next;

}

cout<<endl;

} else {

cout<<"The list is empty.\n";

}

}

};

// test the code

int main() {

LinkedList MyList;

//Add three elements at the end of the list.

MyList.push\_back(10);

MyList.push\_back(20);

MyList.push\_back(30);

MyList.PrintList();

//Delete an element at position 2

MyList.pop\_at(2);

MyList.PrintList();

//Delete an element at position 1

MyList.pop\_at(1);

MyList.PrintList();

return 0;

}