**Lab No. 4: Singly Linked List**

**Objective:**

The objective of this lab is to implement singly linked list.

**Example 1:** This program is an implementation of insertion and display of an array-based one-way linked list.

# include<iostream>

using namespace std;

const int MAX = 20;

int arr[MAX], n; /\* n is the number of elements in the list \*/

int main(){

int choice, item, pos;

while(1){

cout<<"1.Input list\n";

cout<<"2.Insert\n";

cout<<"3.Display\n";

cout<<"4.Quit\n";

cout<<"Enter your choice : ";

cin>>choice;

switch(choice){

case 1:

cout<<"Enter the number of elements to be entered : ";

cin>>n;

input();

break;

case 2:

insert(); break;

case 3:

display(); break;

case 4:

exit(); break;

default:

cout<<"Wrong choice\n";

} /\*End of switch \*/

}/\*End of while \*/

return 0;

}/\*End of main() \*/

input(){

int i;

for(i = 0; i< n ; i++){

cout<<"Input value for element : "<< i+1;

cin>>arr[i];

}

}/\*End of input()\*/

insert(){

int temp,item,position;

if(n == MAX){

cout<<"List overflow\n"; return;

}

cout<<"Enter position for insertion : ";

cin>>position;

cout<<"Enter the value : ";

cin>>item;

if(position > n+1 ){

cout<<"Enter position less than or equal to "<<n+1<<endl;

return;

}

if( position == n+1 ){ /\*Insertion at the end \*/

arr[n] = item;

n = n+1;

}

/\* Insertion in between \*/

temp=n-1;

while( temp >= position-1) {

arr[temp+1] = arr[temp]; /\* shifting right \*/

temp --;

}

arr[position-1] = item;

n = n +1 ;

}/\*End of insert()\*/

display(){

int i;

if(n==0){

cout<<"List is empty\n"; return;

}

for(i = 0; i< n; i++)

cout<<"Value at position "<< i+1<<" "<<arr[i];

}/\*End of display()\*/

**Example 2:** This program implements insertion and display operations for a one-way linked list using pointers.

#include<iostream.h>

#include<conio.h>

using namespace std;

struct Node{

int data;

Node\* next;

};

Node \*first=NULL,\*last=NULL;

void addItem(int d);

void display();

int main(){

addItem(2);

addItem(84);

addItem(6);

display();

getch();

return 0;

}

void addItem(int n){

Node\* ptrNew = new Node;

ptrNew->data = n;

ptrNew->next = NULL;

if (first==NULL){

first = ptrNew;

last = ptrNew;

}

else{

last->next = ptrNew;

last = ptrNew;

}

}

void display(){

Node\* ptrCur = first;

while(ptrCur!=NULL){

cout<<ptrCur->data<<endl;

ptrCur = ptrCur->next;

}

}

**Example 3:** This program implements one-way linked list using classes and pointers.

#include <iostream>

#include <stdlib.h>

using namespace std;

/\* The Node class \*/

class Node

{

public:

int get()

{ return object; }

void set(int object)

{ this->object = object; }

Node \* getNext()

{ return nextNode; }

void setNext(Node \* nextNode)

{ this->nextNode = nextNode; }

private:

int object;

Node \* nextNode;

};

/\* The List class \*/

class List

{

public:

List();

void add (int addObject);

int get();

bool next();

friend void traverse(List list);

friend List addNodes();

private:

int size;

Node \* headNode;

Node \* currentNode;

Node \* lastCurrentNode;

};

/\* Constructor \*/

List::List()

{

headNode = new Node();

headNode->setNext(NULL);

currentNode = NULL;

lastCurrentNode = NULL;

size = 0;

}

/\* add() class method \*/

void List::add (int addObject)

{

Node \* newNode = new Node();

newNode->set(addObject);

if( currentNode != NULL )

{

newNode->setNext(currentNode->getNext());

currentNode->setNext( newNode );

lastCurrentNode = currentNode;

currentNode = newNode;

}

else

{

newNode->setNext(NULL);

headNode->setNext(newNode);

lastCurrentNode = headNode;

currentNode = newNode;

}

size ++;

}

/\* get() class method \*/

int List::get()

{

if (currentNode != NULL)

return currentNode->get();

}

/\* next() class method \*/

bool List::next()

{

if (currentNode == NULL) return false;

lastCurrentNode = currentNode;

currentNode = currentNode->getNext();

if (currentNode == NULL || size == 0)

return false;

else

return true;

}

/\* Friend function to traverse linked list \*/

void traverse(List list)

{

Node\* savedCurrentNode = list.currentNode;

list.currentNode = list.headNode;

for(int i = 1; list.next(); i++)

{

cout << "\n Element " << i << " " << list.get();

}

list.currentNode = savedCurrentNode;

}

/\* Friend function to add Nodes into the list \*/

List addNodes()

{

List list;

list.add(2);

list.add(6);

list.add(8);

list.add(7);

list.add(1);

cout << "\n List size = " << list.size <<'\n';

return list;

}

int main()

{

List list = addNodes();

traverse(list);

return 0;

}

**Exercises:**

1. Write a function to delete the first node from a pointer based linked list.
2. Write a function to delete the last node from a pointer based linked list.
3. Write a function to search a value from a pointer based linked list.
4. Write a function to sort a pointer based one-way linked list of integers.

**Home Work:**

1. Write a function to swap the first and last nodes of a linked list.
2. Write a function to reverse a pointer based one-way linked list