**Mechi Multiple Campus**

(Tribhuvan University)

Bhadrapur, Jhapa



**Lab Report of**

**Data Structures and Algorithm (CACS-201)**

**Implementation of Singly Linked List**

Faculty of Humanities & Social Sciences

Tribhuvan University

Kritipur, Nepal

**Submitted By**

**Name:** Santosh Bhandari

**Roll No:** 58

**Submitted To**

Mechi Multiple Campus

Department of Bachelor in Computer Application

Bhadrapur, Jhapa, Nepal

**Program Code**

#include<stdio.h>

#include<malloc.h>

#include<process.h>

struct node{

int info;

struct node \*next;

};

struct node \*start=NULL;

void insert\_first(int);

void insert\_last(int);

void insert\_specposition(int);

void delete\_first();

void delete\_last();

void delete\_specposition();

void display();

void search();

void main(){

int data,ch;

top:

printf("\nMenu for the program\n");

printf("\n1 to insert data as the first node");

printf("\n2 to insert data as the last node");

printf("\n3 to insert data at the specified position");

printf("\n4 to delete the first node");

printf("\n5 to delete the last node");

printf("\n6 to delete the specified node");

printf("\n7 to display the linked list");

printf("\n8 to search a particular data in the linked list");

printf("\n9 to exit");

printf("\nEnter Your Choice(1,2,3,4,5,6,7,8,9): ");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\nInput the data to insert:");

scanf("%d",&data);

insert\_first(data);

goto top;

case 2:

printf("\nInput the data to insert:");

scanf("%d",&data);

insert\_last(data);

goto top;

case 3:

printf("\nInput the data to insert:");

scanf("%d",&data);

insert\_specposition(data);

goto top;

case 4:

delete\_first();

goto top;

case 5:

delete\_last();

goto top;

case 6:

delete\_specposition();

goto top;

case 7:

display();

goto top;

case 8:

search();

goto top;

case 9:

exit(0);

default:

printf("\nInvalid Choice");

goto top;

}

}

void insert\_first(int data){

struct node \*newnode;

newnode=(struct node \*)malloc(sizeof(struct node));

if(newnode==NULL){

printf("\nOut of memory space\n");

}else{

newnode->info=data;

if(start==NULL){

newnode->next=NULL;

start=newnode;

}else{

newnode->next=start;

start=newnode;

}

printf("\n%d is successfully inserted as the first node\n",data);

}

}

void insert\_last(int data){

struct node \*newnode, \*last;

newnode=(struct node \*)malloc(sizeof(struct node));

if(newnode==NULL){

printf("\nOut of memory space");

}else{

newnode->info=data;

if(start==NULL){

newnode->next=NULL;

start=newnode;

}else{

newnode->next=NULL;

last=start;

while(last->next!=NULL){

last=last->next;

}

last->next=newnode;

}

printf("\n%d is successfully inserted as the last node\n",data);

}

}

void insert\_specposition(int data){

int pos,i;

struct node \*newnode,\*previous;

newnode=(struct node \*)malloc(sizeof(struct node));

if(newnode==NULL){

printf("\nOut of memory space");

}else{

newnode->info=data;

printf("\nInput the position of the node to inset data:");

scanf("%d",&pos);

if(pos==1){

if(start==NULL){

newnode->next=NULL;

start=newnode;

}else{

newnode->next=start;

start=newnode;

}

}else{

previous=start;

i=1;

while(i<pos-1){

previous=previous->next;

if(previous==NULL){

printf("\nSorry insert position in between existing nodes");

return;

}

}

newnode->next=previous->next;

previous->next=newnode;

}

printf("\n%d is successfully inserted at the %d position",data,pos);

}

}

void delete\_first(){

struct node \*temp;

if(start==NULL){

printf("\nList is empty \n");

}else{

temp=start;

start=temp->next;

printf("\nThe deleted element is %d\n",temp->info);

free(temp);

}

}

void delete\_last(){

struct node \*temp, \*last;

if(start==NULL){

printf("\nList is empty \n");

}else{

temp=start;

last=start;

while(temp->next!=NULL) {

last=temp;

temp=temp->next;

}

last->next=NULL;

printf("\nThe deleted element is %d\n",temp->info);

free(temp);

}

}

void delete\_specposition(){

int pos,i;

struct node \*temp, \*previous;

if(start==NULL){

printf("\nList is empty \n");

}else{

printf("\nInput the position of the data to delete\n");

scanf("%d",&pos);

if(pos==1){

temp=start;

start=start->next;

printf("\nThe deleted data is %d\t",temp->info);

free(temp);

}else{

i=1;

temp=start;

while(i<=pos-1){

previous=temp;

temp=temp->next;

if(temp==NULL){

printf("\nSorry insert position in between existing nodes");

return;

}

i=i+1;

}

previous->next=temp->next;

printf("\nThe deleted element is %d\n",temp->info);

free(temp);

}

}

}

void display(){

struct node \*ptr;

if(start==NULL){

printf("\nThe list is empty");

}else{

ptr=start;

printf("\nThe elements of the linked list are:\n");

while(ptr!=NULL){

printf("%d\t",ptr->info);

ptr=ptr->next;

}

}

}

void search(){

int el,search=0;

struct node \*temp;

if(start==NULL){

printf("\nThe list is empty");

}else{

printf("\nInput the data to search:");

scanf("%d",&el);

temp=start;

while(temp!=NULL){

if(temp->info==el){

search=1;

break;

}

temp=temp->next;

}

if(search==1){

printf("\nSearch is successful");

}else{

printf("\nSearch is unsuccessful");

}

}

}

**Output of the Program**



