Experiment No: 4 Date: 23-11-2021

Aim: To perform insertion, deletion, searching and traversal on a singly linked list

Algorithm:

1. START
2. Display a menu containing operations
3. If choice is 1

3.1. Insert a node at beginning of list

3.2. Go to step 2

1. If choice is 2

4.1. Insert a node at end of the list

4.2. Go to step 2

1. If choice is 3

5.1. Insert a node at a given position POS

5.2. Go to step 2

1. If choice is 4

6.1. Delete the node at beginning , position of head

6.2. Go to step 2

1. If choice is 5

7.1. Delete the node at last position

7.2. Go to step 2

1. If choice is 6

8.1. Delete the node residing at a position POS

8.2. Go to step 2

1. If choice is 7

9.1. Input a value to be searched

9.2. Traverse the list till the value found

9.2.1. If it is found, print ‘ Value found at POS’

9.2.2. Else print ‘ Item not found in list’

9.3. Go to step 2

1. If choice is 8, print all items in the list
2. STOP

Program code:

#include<stdio.h>

#include<stdlib.h>

int count=0;

void display();

void searchelement();

void insert\_begin();

void insert\_end();

void insert\_pos();

void deletebegin();

void deleteend();

void deletepos();

struct node

{

int data;

struct node \*next;

}\*head=NULL;

int main()

{

int opt;

do

{

printf("\n SELECT ANY OF THE OPTIONS FROM THE BELOW LIST\n");

printf("\n1. INSERTION at beginning\n");

printf("\n2. INSERTION at end\n");

printf("\n3. INSERTION at a Position\n");

printf("\n4. DELETION at beginning\n");

printf("\n5. DELETION at end\n");

printf("\n6. DELETION at a position\n");

printf("\n7. SEARCH AN ELEMENT\n");

printf("\n8. TRAVERSAL\n");

printf("\n9. EXIT\n");

scanf("%d",&opt);

switch(opt)

{

case 1: insert\_begin();

break;

case 2: insert\_end();

break;

case 3: insert\_pos();

break;

case 4: deletebegin();

break;

case 5: deleteend();

break;

case 6: deletepos();

break;

case 7: searchelement();

break;

case 8: display();

break;

case 9: exit(0);

default: printf("\n Invalid Choice\n");

}

}

while(opt!=8);

return 0;

}

void insert\_end()

{

struct node \*newnode;

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

printf("\n enter a value: ");

scanf("%d",&newnode->data);

if(head==NULL)

head=newnode;

else

{

struct node \*temp=head;

while(temp->next!=NULL)

temp=temp->next;

temp->next=newnode;

newnode->next=NULL;

}

count++;

if(head!=NULL)

{

printf("\n The current values in the linked list are\n");

display();

}

}

void insert\_begin()

{

struct node \*newnode;

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

printf("\n enter a value: ");

scanf("%d",&newnode->data);

if(head==NULL)

{

head=newnode;

newnode->next=NULL;

}

else

{

newnode->next=head;

head=newnode;

}

count++;

if(head!=NULL)

{

printf("\n current items in the linked list are:\n");

display();

}

}

void insert\_pos()

{

int pos,i=1;

struct node \*newnode;

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

printf("\n Enter a position to which a node is to be inserted: ");

scanf("%d",&pos);

printf("\n enter a value: ");

scanf("%d",&newnode->data);

if(pos==1)

{

newnode->next=head;

head=newnode;

display();

}

else

{

struct node \*temp=head, \*prev;

for(i=1;i<pos && temp!=NULL;i++)

{

prev = temp;

temp = temp->next;

}

newnode->next = prev->next;

prev->next = newnode;

if(head!=NULL)

{

printf("\n current items in the list are\n");

display();

}

}

}

void deletebegin()

{

if(head==NULL)

printf("\n Linked list is already empty\n");

else

{

struct node \*temp=head;

head=temp->next;

free(temp);

count--;

printf("\n current elements in the list are\n");

display();

}

}

void deleteend()

{

if(head==NULL)

printf("\n the list is already empty...\n");

else

{

struct node \*temp=head;

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

temp=temp->next;

free(temp->next->next);

temp->next=NULL;

count--;

printf("\n the current elements in the list are\n");

display();

}

}

void deletepos()

{

int pos,i=1;

if(head==NULL)

printf("\n The list is already empty...!\n");

else

{

printf("\n enter a position in which the node is to be deleted: ");

scanf("%d",&pos);

if(pos<=count)

{

if(pos!=count)

{

struct node \*temp=head;

struct node \*temp1=temp;

while(i!=pos)

{

temp1=temp;

temp=temp->next;

i++;

}

temp1->next=temp->next;

free(temp);

count--;

}

else

{

struct node \*temp=head;

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

temp=temp->next;

free(temp->next->next);

temp->next=NULL;

}

}

else

printf("\n Position %d not within the list\n",pos);

if(head!=NULL)

{

printf("\n the current items in the list are\n");

display();

}

}

}

void display()

{

struct node \*temp=head;

if(head==NULL)

printf("\n there is no item to show....\n");

else

{

//struct node \*temp=head;

while(temp!=NULL)

{

printf("%d\t",temp->data);

temp=temp->next;

}

}

}

void searchelement()

{

struct node \*temp=head;

int key;

int i=1;

printf("\n enter a value to be searched: ");

scanf("%d",&key);

while(temp->data!=key&&temp->next!=NULL)

{

temp=temp->next;

i++;

}

if(temp->data==key)

printf("\n the key is found in the list at position %d",i);

else

printf("\n key not found!..\n");

}#include<stdio.h>

#include<stdlib.h>

int count=0;

void display();

void searchelement();

void insert\_begin();

void insert\_end();

void insert\_pos();

void deletebegin();

void deleteend();

void deletepos();

struct node

{

int data;

struct node \*next;

}\*head=NULL;

int main()

{

int opt;

do

{

printf("\n SELECT ANY OF THE OPTIONS FROM THE BELOW LIST\n");

printf("\n1. INSERTION at beginning\n");

printf("\n2. INSERTION at end\n");

printf("\n3. INSERTION at a Position\n");

printf("\n4. DELETION at beginning\n");

printf("\n5. DELETION at end\n");

printf("\n6. DELETION at a position\n");

printf("\n7. SEARCH AN ELEMENT\n");

printf("\n8. TRAVERSAL\n");

printf("\n9. EXIT\n");

scanf("%d",&opt);

switch(opt)

{

case 1: insert\_begin();

break;

case 2: insert\_end();

break;

case 3: insert\_pos();

break;

case 4: deletebegin();

break;

case 5: deleteend();

break;

case 6: deletepos();

break;

case 7: searchelement();

break;

case 8: display();

break;

case 9: exit(0);

default: printf("\n Invalid Choice\n");

}

}

while(opt!=8);

return 0;

}

void insert\_end()

{

struct node \*newnode;

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

printf("\n enter a value: ");

scanf("%d",&newnode->data);

if(head==NULL)

head=newnode;

else

{

struct node \*temp=head;

while(temp->next!=NULL)

temp=temp->next;

temp->next=newnode;

newnode->next=NULL;

}

count++;

if(head!=NULL)

{

printf("\n The current values in the linked list are\n");

display();

}

}

void insert\_begin()

{

struct node \*newnode;

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

printf("\n enter a value: ");

scanf("%d",&newnode->data);

if(head==NULL)

{

head=newnode;

newnode->next=NULL;

}

else

{

newnode->next=head;

head=newnode;

}

count++;

if(head!=NULL)

{

printf("\n current items in the linked list are:\n");

display();

}

}

void insert\_pos()

{

int pos,i=1;

struct node \*newnode;

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

printf("\n Enter a position to which a node is to be inserted: ");

scanf("%d",&pos);

printf("\n enter a value: ");

scanf("%d",&newnode->data);

if(pos==1)

{

newnode->next=head;

head=newnode;

display();

}

else

{

struct node \*temp=head, \*prev;

for(i=1;i<pos && temp!=NULL;i++)

{

prev = temp;

temp = temp->next;

}

newnode->next = prev->next;

prev->next = newnode;

if(head!=NULL)

{

printf("\n current items in the list are\n");

display();

}

}

}

void deletebegin()

{

if(head==NULL)

printf("\n Linked list is already empty\n");

else

{

struct node \*temp=head;

head=temp->next;

free(temp);

count--;

printf("\n current elements in the list are\n");

display();

}

}

void deleteend()

{

if(head==NULL)

printf("\n the list is already empty...\n");

else

{

struct node \*temp=head;

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

temp=temp->next;

free(temp->next->next);

temp->next=NULL;

count--;

printf("\n the current elements in the list are\n");

display();

}

}

void deletepos()

{

int pos,i=1;

if(head==NULL)

printf("\n The list is already empty...!\n");

else

{

printf("\n enter a position in which the node is to be deleted: ");

scanf("%d",&pos);

if(pos<=count)

{

if(pos!=count)

{

struct node \*temp=head;

struct node \*temp1=temp;

while(i!=pos)

{

temp1=temp;

temp=temp->next;

i++;

}

temp1->next=temp->next;

free(temp);

count--;

}

else

{

struct node \*temp=head;

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

temp=temp->next;

free(temp->next->next);

temp->next=NULL;

}

}

else

printf("\n Position %d not within the list\n",pos);

if(head!=NULL)

{

printf("\n the current items in the list are\n");

display();

}

}

}

void display()

{

struct node \*temp=head;

if(head==NULL)

printf("\n there is no item to show....\n");

else

{

//struct node \*temp=head;

while(temp!=NULL)

{

printf("%d\t",temp->data);

temp=temp->next;

}

}

}

void searchelement()

{

struct node \*temp=head;

int key;

int i=1;

printf("\n enter a value to be searched: ");

scanf("%d",&key);

while(temp->data!=key&&temp->next!=NULL)

{

temp=temp->next;

i++;

}

if(temp->data==key)

printf("\n the key is found in the list at position %d",i);

else

printf("\n key not found!..\n");

}

OUTPUT



