Linked list

yapara karthikeya

1BM21CS249

Code:

#include<stdio.h>

#include<stdlib.h>

struct NODE

{

int data;

struct NODE \*link;

};

typedef struct NODE node;

node \*start;

void create()

{

int ch;

node \*new,\*curr=NULL;

start=NULL;

start=(node \*)malloc(sizeof(node));

curr=start;

printf("Enter the element:");

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

while(1)

{

printf("Add another element (1 for yes 0 for no)?");

scanf("%d",&ch);

if(ch)

{

new=(node \*)malloc(sizeof(node));

printf("Enter the element:");

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

curr->link=new;

curr=new;

}

else

{

curr->link=NULL;

break;

}

}

}

void insert\_beg()

{

node \*new;

new=(node \*)malloc(sizeof(node));

printf("\nEnter element:");

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

if(start==NULL)

{

start=new;

new->link=NULL;

return;

}

new->link=start;

start=new;

}

void insert\_end()

{

node \*new,\*temp;

new=(node \*)malloc(sizeof(node));

printf("\nEnter element:");

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

if(start==NULL)

{

start=new;

new->link=NULL;

return;

}

temp=start;

while(temp->link!=NULL)

temp=temp->link;

temp->link=new;

new->link=NULL;

}

void insert\_pos()

{

node \*new,\*temp;

int pos,i=0;

new=(node \*)malloc(sizeof(node));

printf("\nEnter element:");

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

printf("\nEnter position:");

scanf("%d",&pos);

if(pos==1)

{

new->link=start;

start=new;

return;

}

temp=start;

while(i<(pos-1)&&temp->link!=NULL)

{

temp=temp->link;

i++;

}

if(i==(pos-1))

{

new->link=temp->link;

temp->link=new;

return;

}

if(temp==NULL)

printf("\nInvalid Position");

}

void display()

{

node \*temp;

if(start==NULL)

{

printf("Linked list is empty");

return;

}

temp=start;

while(temp!=NULL)

{

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

temp=temp->link;

}

}

void main()

{

int choice;

while(1)

{

printf("1.Create 2.Insert at the beginning 3.Insert at the end

4.Insert at a given position 5.Display 6.Exit\n");

printf("Enter your choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:create();

break;

case 2:insert\_beg();

break;

case 3:insert\_end();

break;

case 4:insert\_pos();

break;

case 5:display();

break;

case 6:exit(0);

break;

default:printf("Wrong choice\n");

}

}

}

For Delete

#include<stdio.h>

#include<stdlib.h>

struct NODE

{

int data;

struct NODE \*link;

};

typedef struct NODE node;

node \*start;

void create()

{

int ch;

node \*new,\*curr=NULL;

start=NULL;

start=(node \*)malloc(sizeof(node));

curr=start;

printf("Enter the element:");

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

while(1)

{

printf("Add another element (1 for yes 0 for no)?");

scanf("%d",&ch);

if(ch)

{

new=(node \*)malloc(sizeof(node));

printf("Enter the element:");

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

curr->link=new;

curr=new;

}

else

{

curr->link=NULL;

break;

}

}

}

void del\_beg()

{

node \*temp;

if(start==NULL)

{

printf("Linked list is empty");

return;

}

temp=start;

start=start->link;

free(temp);

}

void del\_end()

{

node \*next,\*temp;

if(start==NULL)

{

printf("Linked list is empty");

return;

}

if(start->link==NULL)

{

free(start);

start=NULL;

return;

}

temp=start;

next=start->link;

while(next->link!=NULL)

{

temp=next;

next=next->link;

}

free(next);

temp->link=NULL;

}

void del\_ele()

{

node \*prev,\*curr;

int ele;

if(start==NULL)

{

printf("Linked list is empty");

return;

}

if(start->link==NULL)

{

ele=start->data;

free(start);

start=NULL;

return;

}

printf("Enter element:");

scanf("%d",&ele);

prev=start;

curr=start->link;

while(curr->data!=ele && curr!=NULL)

{

prev=curr;

curr=curr->link;

}

if(curr->data==ele)

{

prev->link=curr->link;

free(curr);

return;

}

printf("Element not found");

}

void display()

{

node \*temp;

if(start==NULL)

{

printf("Linked list is empty");

return;

}

temp=start;

while(temp!=NULL)

{

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

temp=temp->link;

}

}

void main()

{

int choice;

while(1)

{

printf("\n1.Create 2.Delete at the beginning 3.Delete at the end

4.Delete a given element 5.Display 6.Exit\n");

printf("Enter your choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:create();

break;

case 2:del\_beg();

break;

case 3:del\_end();

break;

case 4:del\_ele();

break;

case 5:display();

break;

case 6:exit(0);

break;

default:printf("Wrong choice\n");

}

}

}

Output: