**Doubly Linked list and various operations.**

#include <iostream>

#include<string>

using namespace std;

typedef struct linked{

int data;

linked \*next,\*prev;

};

linked\* create(linked \*h){

int cnt;

cout<<endl<<"Enter number of nodes:";

cin>>cnt;

linked \*head=h,\*p,\*q;

int a;

head=new linked;

cout<<endl<<"Enter data:";

cin>>a;

head->data=a;

head->next=head->prev=NULL;

p=q=head;

for(int i=1;i<cnt;i++){

q->next=new linked;

q=q->next;

cout<<endl<<"Enter data:";

cin>>a;

q->data=a;

p->next=q;

q->next=NULL;

p=q;

}

return head;

}

void searchs(linked \*h){

linked \*p=h;

int key,flag=0;

cout<<endl<<"Enter element do you want to search:";

cin>>key;

while(p!=NULL){

if(p->data==key){

flag=1;

}

p=p->next;

}

if(flag==1){

cout<<endl<<"Element is found!!";

}else{

cout<<endl<<"Element is not found!!";

}

}

void display(linked \*h){

linked \*p=h;

cout<<endl<<"Linked list elements:";

while(p!=NULL){

cout<<p->data<<"\t";

p=p->next;

}

}

void reverses(linked \*h){

if(h!=NULL){

reverses(h->next);

cout<<h->data<<"\t";

}

}

void counts(linked \*h){

linked \*p=h;

int cnt=0;

while(p!=NULL){

cnt++;

p=p->next;

}

cout<<endl<<"Number of nodes:"<<cnt;

}

linked\* addBeg(linked \*head){

linked \*q,\*p;

int x;

cout<<endl<<"Enter data:";

cin>>x;

q=new linked;

q->data=x;

q->next=q->prev=NULL;

if(head == NULL)

{

head=q;

return(head);

}

else

{

q->next=head;

head->prev=q;

q->prev=NULL;

head=q;

return(head);

}

}

linked\* addMid(linked \*head){

linked \*q,\*p;

int x,loc;

cout<<endl<<"Enter data:";

cin>>x;

q=new linked;

q->data=x;

q->next=q->prev=NULL;

if(head == NULL)

{

head=q;

return(head);

}

else

{

p=head;

cout<<endl<<"Enter location:";

cin>>loc;

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

p=p->next;

}

q->next=p->next;

(q->next)->prev=p;

p->next=q;

q->prev=p;

return(head);

}

}

linked\* addEnd(linked \*head){

linked \*q,\*p;

int x;

cout<<endl<<"Enter data:";

cin>>x;

q=new linked;

q->data=x;

q->next=q->prev=NULL;

if(head == NULL)

{

head=q;

return(head);

}

else

{

p=head;

while(p->next!=NULL){

p=p->next;

}

q->prev=p;

p->next=q;

return(head);

}

}

linked\* delBeg(linked \*h){

linked \*head,\*p,\*q;

head=h;

if(head==NULL){

cout<<endl<<"List is Empty..";

}

else{

head=head->next;

head->prev=NULL;

return head;

}

}

linked\* delMid(linked \*h){

linked \*head,\*p,\*q;

head=h;

int loc;

if(head==NULL){

cout<<endl<<"List is Empty..";

}

else{

p=head;

cout<<endl<<"Enter location:";

cin>>loc;

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

p=p->next;

}

q=p->next;

p->next=q->next;

(q->next)->prev=p;

delete q;

return head;

}

}

linked\* delEnd(linked \*h){

linked \*head,\*p,\*q;

head=h;

if(head==NULL){

cout<<endl<<"List is Empty..";

}

else{

p=head;

while((p->next)->next!=NULL){

p=p->next;

}

q=p->next;

p->next=NULL;

q->prev=NULL;

delete q;

return head;

}

}

int main()

{

linked \*head;

int ch;

do{

cout<<endl<<"\*\*\*\*\*DOUBLY LINKED LIST OPERATIONS\*\*\*\*\*\*";

cout<<endl<<"1.Create list...\n2.Display list...\n3.Insert node at the beginning...\n4.Insert node at the middle...\n5.Insert node at the end...\n6.Delete first element...\n7.Delete middle element...\n8.Delete last element...\n9.search particular element..\n10.Count number of nodes...\n11.Reverse elements..\n12.Exit..";

cout<<endl<<"Enter your choice:";

cin>>ch;

switch(ch){

case 1:

head=create(head);

break;

case 2:

display(head);

break;

case 3:

head=addBeg(head);

break;

case 4:

head=addMid(head);

break;

case 5:

head=addEnd(head);

break;

case 6:

head=delBeg(head);

break;

case 7:

head=delMid(head);

break;

case 8:

head=delEnd(head);

break;

case 9:

searchs(head);

break;

case 10:

counts(head);

break;

case 11:

reverses(head);

break;

case 12:

break;

default:

cout<<endl<<"Wrong choice entered....";

}

}while(ch!=10);

return 0;

}