* SINGLE LINKED LIST

#include<stdio.h>

#include<stdlib.h>

struct node\* create();

void display (struct node\*);

struct node \* getNode();

struct node \* insert(struct node \*);

struct node \* delete(struct node \*);

struct node \* revList(struct node \*);

struct node

{

int data;

struct node \*next;

};

int main()

{

int ch;

struct node \*head;

while(1)

{

printf("1.create\n2.display\n3.insert\n4.delete\n5.reverse\n6.exit");

printf("\nEnter your choice");

scanf("%d",&ch);

switch(ch)

{

case 1: head=create();

break;

case 2: display(head);

break;

case 3: head=insert(head);

break;

case 4: head=delete(head);

break;

case 5: head=revList(head);

display(head);

break;

case 6: exit(0);

break;

default :

printf("Eneter Valid choice");

break;

}

}

}

struct node \*create()

{

struct node \*head,\*first,\*newnode;

first =getNode();

head=first;

newnode = getNode();

printf("Enter the value");

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

newnode->next=NULL;

while(newnode->data!=-9)

{

first->next=newnode;

first=newnode;

newnode =getNode();

printf("Enter the value");

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

newnode->next=NULL;

}

return head->next;

}

void display(struct node \*head)

{

struct node \*temp;

temp=head;

while(temp!=NULL)

{

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

temp=temp->next;

}

}

struct node \*getNode()

{

struct node \*np;

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

np->data=0;

np->next=NULL;

return np;

}

struct node \* insert(struct node \*head)

{

struct node \*newnode,\*temp;

int ch,pos,i;

temp=head;

newnode=getNode();

printf("Enter the value");

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

printf("\n1.Insert at Beg\n2.Insert at end\n3.Insert at pos");

scanf("%d",&ch);

switch(ch)

{

case 1:

newnode->next=head;

head=newnode;

return head;

break;

case 2:

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

return head;

break;

case 3:

printf("Enter poss");

scanf("%d",&pos);

for(i=1;i<pos-1;i++)

{

temp=temp->next;

}

newnode->next=temp->next;

temp->next=newnode;

return head;

break;

default:

printf("Enter valid choice");

break;

}

}

struct node \* delete(struct node \*head)

{

struct node \* temp;

int ch,pos,i;

temp=head;

printf("\n1.Delete at Beg\n2.Delete at end\n3.Delete at pos");

scanf("%d",&ch);

switch(ch)

{

case 1 :

head=temp->next;

return head;

break;

case 2 :

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

{

temp=temp->next;

}

temp->next=NULL;

return head;

break;

case 3 :

printf("Enter poss");

scanf("%d",&pos);

for(i=1;i<pos-1;i++)

{

temp=temp->next;

}

temp->next=temp->next->next;

return head;

break;

default : printf("Invalid choice");

break;

}

}

struct node \* revList(struct node \*head)

{

struct node \*temp,\*rhead,\*newnode;

temp=head;

rhead=getNode();

rhead->data=temp->data;

newnode=getNode();

while (temp->next!=NULL)

{

temp=temp->next;

newnode->data=temp->data;

newnode->next= rhead;

rhead=newnode;

newnode=getNode();

}

return rhead;

}

* DOUBLE LINKED LIST

#include<stdio.h>

#include<stdlib.h>

struct node \*create();

struct node \*getnode();

void display(struct node \*);

struct node \*insert(struct node \*);

struct node \*delete(struct node \*);

struct node{

int data;

struct node \*next;

struct node \*prevs;

};

int main()

{

int ch;

struct node \*head;

while(1)

{

printf("1.create\n2.display\n3.insert\n4.delete\n5.exit");

printf("\nEnter your choice");

scanf("%d",&ch);

switch(ch)

{

case 1: head=create();

break;

case 2:display(head);

break;

case 3: head=insert(head);

break;

case 4: head=delete(head);

break;

case 5: exit(0);

break;

default :

printf("Eneter Valid choice");

break;

}

}

}

struct node \*getnode()

{

struct node \*np;

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

np->data=0;

np->next=NULL;

np->prevs=NULL;

return np;

}

struct node \*create()

{

struct node \*head,\*first,\*newnode;

first=getnode();

head=first;

newnode=getnode();

printf("Enter the value");

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

while(newnode->data!=-9)

{

newnode->prevs=first;

first->next=newnode;

first=newnode;

newnode=getnode();

printf("Enter the value");

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

}

return head->next;

}

void display(struct node \*head)

{

struct node \*temp;

temp=head;

while(temp!=NULL)

{

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

temp=temp->next;

}

}

struct node \* insert(struct node \*head)

{

struct node \*newnode,\*temp;

int ch,pos,i;

temp=head;

newnode=getnode();

printf("Enter the value");

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

printf("\n1.Insert at Beg\n2.Insert at end\n3.Insert at pos");

scanf("%d",&ch);

switch(ch)

{

case 1:

newnode->next=head;

head->prevs=newnode;

head=newnode;

return head;

break;

case 2:

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

newnode->prevs=temp;

return head;

break;

case 3:

printf("Enter poss");

scanf("%d",&pos);

for(i=1;i<pos-1;i++)

{

temp=temp->next;

}

newnode->prevs=temp;

newnode->next=temp->next;

temp->next=newnode;

temp->next->prevs=newnode;

return head;

break;

default:

printf("Enter valid choice");

break;

}

}

struct node \* delete(struct node \*head)

{

struct node \* temp;

int ch,pos,i;

temp=head;

printf("\n1.Delete at Beg\n2.Delete at end\n3.Delete at pos");

scanf("%d",&ch);

switch(ch)

{

case 1 :

head=temp->next;

temp->prevs=NULL;

return head;

break;

case 2 :

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

{

temp=temp->next;

}

temp->next->prevs=NULL;

temp->next=NULL;

return head;

break;

case 3 :

printf("Enter poss");

scanf("%d",&pos);

for(i=1;i<pos-1;i++)

{

temp=temp->next;

}

temp->next=temp->next->next;

temp->next->prevs=temp;

return head;

break;

default : printf("Invalid choice");

break;

}

}

* CIRCULAR LINKEDLIST

#include<stdio.h>

#include<stdlib.h>

struct node\* create();

void display (struct node\*);

struct node \* getNode();

struct node \* insert(struct node \*);

struct node \* delete(struct node \*);

struct node \* revList(struct node \*);

struct node

{

int data;

struct node \*next;

};

int main()

{

int ch;

struct node \*head;

head=NULL;

while(1)

{

printf("1.create\n2.display\n3.insert\n4.delete\n5.reverse\n6.exit");

printf("\nEnter your choice");

scanf("%d",&ch);

switch(ch)

{

case 1: head=create();

break;

case 2: display(head);

break;

case 3: head=insert(head);

break;

case 4: head=delete(head);

break;

case 5: head=revList(head);

display(head);

break;

case 6: exit(0);

break;

default :

printf("Eneter Valid choice");

break;

}

}

}

struct node \*create()

{

struct node \*head,\*first,\*newnode;

first =getNode();

head=first;

newnode = getNode();

printf("Enter the value");

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

while(newnode->data!=-9)

{

first->next=newnode;

first=newnode;

newnode =getNode();

printf("Enter the value");

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

newnode->next=head->next;

}

return head->next;

}

void display(struct node \*head)

{

struct node \*temp;

temp=head;

if(head==NULL)

{

printf("List is empty");

}

while(temp->next!=head)

{

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

temp=temp->next;

}

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

}

struct node \*getNode()

{

struct node \*np;

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

np->data=0;

np->next=NULL;

return np;

}

struct node \* insert(struct node \*head)

{

struct node \*newnode,\*temp;

int ch,pos,i;

temp=head;

newnode=getNode();

printf("Enter the value");

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

printf("\n1.Insert at Beg\n2.Insert at end\n3.Insert at pos");

scanf("%d",&ch);

switch(ch)

{

case 1:

while(temp->next!=head)

{

temp=temp->next;

}

newnode->next=head;

temp->next=newnode;

head=newnode;

return head;

break;

case 2:

while(temp->next!=head)

{

temp=temp->next;

}

newnode->next=head;

temp->next=newnode;

return head;

break;

case 3:

printf("Enter poss");

scanf("%d",&pos);

if(pos==1)

{

while(temp->next!=head)

{

temp=temp->next;

}

newnode->next=head;

temp->next=newnode;

head=newnode;

return head;

break;

}

else

{

for(i=1;i<pos-1;i++)

{

temp=temp->next;

}

newnode->next=temp->next;

temp->next=newnode;

return head;

break;

}

default:

printf("Enter valid choice");

break;

}

}

struct node \* delete(struct node \*head)

{

struct node \* temp;

int ch,pos,i;

temp=head;

if(temp==NULL)

{

printf("List is empty");

}

else

{

printf("\n1.Delete at Beg\n2.Delete at end\n3.Delete at pos");

scanf("%d",&ch);

switch(ch)

{

case 1 :

if(head->next==head)

{

head=NULL;

return head;

}

else

{

while(temp->next!=head)

{

temp=temp->next;

}

//need to check

temp=head->next;

head=head->next;

return head;

}

case 2 :

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

{

temp=temp->next;

}

temp->next=head;

return head;

break;

case 3 :

printf("Enter poss");

scanf("%d",&pos);

for(i=1;i<pos-1;i++)

{

temp=temp->next;

}

temp->next=temp->next->next;

return head;

break;

default : printf("Invalid choice");

break;

}

}

}

struct node \* revList(struct node \*head)

{

struct node \*temp,\*rhead,\*newnode;

temp=head;

rhead=getNode();

rhead->data=temp->data;

newnode=getNode();

while (temp->next!=NULL)

{

temp=temp->next;

newnode->data=temp->data;

newnode->next= rhead;

rhead=newnode;

newnode=getNode();

}

return rhead;

}