NAME: Arundarasi Rajendran

PRN:18070122081

BATCH: C4

**CIRCULAR LINKED LIST**

**CREATE+DISPLAY+INSERT+DELETE + REVERSE + LENGTH + SEARCH +COPY**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

};

class list:public Node

{

Node \*listptr,\*temp,\*listptr2,\*temp2;

public:

list()

{

listptr=NULL;

temp=NULL;

listptr2=NULL;

temp2=NULL;

}

void createmore();

void displayall();

void insert\_first(int n);

void insert\_last(int n);

void insert\_between(int n);

void delete\_start();

void delete\_last();

void delete\_between();

void reverse();

void length();

int search(int n);

void copyfirst();

void copyall(int n);

};

int main()

{

list l;

int c,i,n,k;

while(1)

{

cout<<"\n1.Create nodes\n2.Display all terms\n3.Insert at first\n4.Insert at last\n5.Insert inbetween\n6.Delete at start\n7.Delete at last\n8.Delete between\n9.Length\n10.Reverse\n11.Search an element\n12.Copy all nodes\nExit";

cout<<"\nEnter your choice:\n";

cin>>c;

switch(c)

{

case 1:cout<<"Enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

l.createmore();

}

break;

case 2:l.displayall();

break;

case 3:cout<<"Enter element to enter at start of list:";

cin>>n;

l.insert\_first(n);

break;

case 4:cout<<"Enter element to enter at end of list:";

cin>>n;

l.insert\_last(n);

break;

case 5:cout<<"Enter element inbetween list:";

cin>>n;

l.insert\_between(n);

break;

case 6:l.delete\_start();

break;

case 7:l.delete\_last();

break;

case 8:l.delete\_between();

break;

case 9:l.length();

break;

case 10:l.reverse();

break;

case 11:cout<<"Enter an element you want to search in circular linked list:";

cin>>n;

i=l.search(n);

cout<<"index="<<i;

case 12:l.copyall(k);

break;

case 16:return 0;

}

}

return 0;

}

void list::createmore()

{

Node \*newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(listptr==NULL)

{

listptr=newnode;

newnode->next=listptr;

}

if(listptr!=NULL)

{

temp=listptr;

while(temp->next!=listptr)

{

temp=temp->next;

}

temp->next=newnode;

newnode->next=listptr;

}

}

void list::displayall()

{

Node \*d=listptr;

if(d==NULL)

{

cout<<"List is empty";

}

while(d->next!=listptr)

{

cout<<d->data;

cout<<"-->";

d=d->next;

}

cout<<d->data;

cout<<"-->";

d=d->next;

cout<<d->data;

cout<<"-->";

}

void list::insert\_first(int n)

{

Node \*newnode=new Node;

Node \*temp;

newnode->data=n;

newnode->next=NULL;

temp=listptr;

while(temp->next!=listptr)

{

temp=temp->next;

}

temp->next=newnode;

newnode->next=listptr;

listptr=newnode;

}

void list::insert\_last(int n)

{

Node \*newnode=new Node;

Node \*temp;

temp=listptr;

newnode->data=n;

while(temp->next!=listptr)

{

temp=temp->next;

}

temp->next=newnode;

newnode->next=listptr;

}

void list::insert\_between(int n)

{

Node \*newnode=new Node;

newnode->data=n;

newnode->next=temp->next;

temp->next=newnode;

}

void list::delete\_start()

{

Node \*p=listptr,\*f=listptr,\*temp,\*q;

temp=listptr;

q=temp->next;

if(listptr==NULL)

{

cout<<"empty";

}

if(p->next==p)

{

listptr=NULL;

}

while(p->next!=listptr)

{

p=p->next;

}

p->next=f->next;

listptr=q;

free(f);

}

void list::delete\_last()

{

Node \*p=listptr,\*temp=listptr,\*q;

if(listptr==NULL)

{

cout<<"empty";

}

if(p->next==p)

{

listptr=NULL;

}

while(p->next!=listptr)

{

q=p;

p=p->next;

}

q->next=p->next;

listptr=q->next;

free(p);

}

void list::delete\_between()

{

Node \*q,\*temp;

temp=listptr;

q=temp->next;

temp->next=q->next;

free(q);

}

void list::length()

{

Node \*temp;

temp=listptr;

int count=0;

if (listptr != NULL)

{

do

{

temp = temp->next;

count++;

} while (temp != listptr);

}

cout<<"length="<<count<<endl;

}

void list::reverse()

{

Node \*back,\*curr,\*forw;

forw=listptr;

curr=NULL;

back=NULL;

while(forw!=NULL)

{

back=curr;

curr=forw;

forw=forw->next;

curr->next=back;

}

listptr=curr->next;

}

int list::search(int n)

{

int index = 0;

Node \*current = listptr;

do

{

if (current->data == n)

return index;

current = current->next;

index++;

} while (current != listptr);

return -1;

}

void list::copyall(int n)

{

temp=listptr;

Node \*q;

q=listptr;

int i;

Node \*t;

for(i=0;i<n;i++)

{

Node \*newnode=new Node;

newnode->data=temp->data;

newnode->next=temp->next;

if(q==NULL)

{

q=newnode;

}

temp=temp->next;

}

//display;

t=q;

while(t->next!=listptr)

{

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

t=t->next;

}

cout<<t->data;

cout<<"-->";

cout<<"NULL";

}

**CONCATENATE + SORT + UNION + INTERSECTION**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

class Node

{

public:

int data;

Node \*next;

};

class list:public Node

{

Node \*l1,\*t1,\*l2,\*t2;

public:

list()

{

l1=NULL;

t1=NULL;

l2=NULL;

t2=NULL;

}

void createmoreA();

void displayallA();

void createmoreB();

void displayallB();

void sortA();

void sortB();

void intersect();

void concate();

};

int main()

{

list l;

int c,i,n,k;

while(1)

{

cout<<"\n1.Create more nodes of A\n2.Display all terms of A\n3.Sort linked list A\n4.Create more nodes of B\n5.Display all terms of B\n6.Sort linked list B\n7.Concatenate\nExit";

cout<<"\nEnter your choice:\n";

cin>>c;

switch(c)

{

case 1:cout<<"Enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

l.createmoreA();

}

break;

case 2:l.displayallA();

break;

case 3:l.sortA();

break;

case 4:cout<<"Enter number of nodes:";

cin>>k;

for(i=0;i<k;i++)

{

l.createmoreB();

}

break;

case 5:l.displayallB();

break;

case 6:l.sortB();

break;

case 7:l.concate();

break;

case 8:return 0;

}

}

return 0;

}

void list::createmoreA()

{

Node \*newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(l1==NULL)

{

l1=newnode;

newnode->next=l1;

}

if(l1!=NULL)

{

t1=l1;

while(t1->next!=l1)

{

t1=t1->next;

}

t1->next=newnode;

newnode->next=l1;

}

}

void list::displayallA()

{

Node \*d=l1;

if(d==NULL)

{

cout<<"List is empty";

}

while(d->next!=l1)

{

cout<<d->data;

cout<<"-->";

d=d->next;

}

cout<<d->data;

cout<<"-->";

cout<<"NULL\n";

}

void list::sortA()

{

Node \*temp,\*temp1;

temp=l1;

temp1=NULL;

int t;

while(temp->next!=l1)

{

temp1=temp->next;

while(temp1->next!=l1)

{

if(temp->data>temp1->data)

{

t=temp->data;

temp->data=temp1->data;

temp1->data=t;

}

temp1=temp1->next;

}

cout<<temp->data<<"-->";

temp=temp->next;

}

cout<<"NULL";

}

void list::createmoreB()

{

Node \*newnode=new Node;

int n;

cout<<"Enter an element:";

cin>>n;

newnode->data=n;

newnode->next=NULL;

if(l2==NULL)

{

l2=newnode;

newnode->next=l2;

}

if(l1!=NULL)

{

t2=l2;

while(t2->next!=l2)

{

t2=t2->next;

}

t2->next=newnode;

newnode->next=l2;

}

}

void list::displayallB()

{

Node \*d=l2;

if(d==NULL)

{

cout<<"List is empty";

}

while(d->next!=l2)

{

cout<<d->data;

cout<<"-->";

d=d->next;

}

cout<<d->data;

cout<<"-->";

cout<<"NULL\n";

}

void list::sortB()

{

Node \*temp,\*temp1;

temp=l2;

temp1=NULL;

int t;

while(temp->next!=l2)

{

temp1=temp->next;

while(temp1->next!=l2)

{

if(temp->data>temp1->data)

{

t=temp->data;

temp->data=temp1->data;

temp1->data=t;

}

temp1=temp1->next;

}

cout<<temp->data<<"-->";

temp=temp->next;

}

cout<<"NULL";

}

void list::concate()

{

Node \*head1,\*head2,\*p,\*r;

head1=l1;

head2=l2;

p=head1;

while(p->next!=l1)

{

p=p->next;

}

p->next=head2;

r=head2;

while(r->next!=l2)

{

r=r->next;

}

r->next=head1;

cout<<"concatenate linked list:\n";

Node \*newnode=l1;

if(newnode==NULL)

{

cout<<"List is empty";

}

while(newnode->next!=l1)

{

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

}

cout<<newnode->data;

cout<<"-->";

newnode=newnode->next;

cout<<newnode->data;

}

<https://codeforwin.org/2018/11/c-program-to-reverse-a-circular-linked-list.html>