doubly\_07.cpp

**Compile:** g++ doubly\_07.cpp -o doubly\_07

**Run:** ./doubly\_07

**Program:**

#include<iostream>

#include<stdlib.h>

int ch,data1,data2;

using namespace std;

struct node

{

struct node \*lptr;

int data;

struct node \*rptr;

};

struct node \*list=NULL,\*p,\*q,\*r,\*s;

class doubly

{

public:

void menu()

{

do

{

cout<<endl<<"1.Insert at the begin \n 2. Display \n 3.Exit \n 4.Insert at the end \n 5.Insert before a particular node \n 6. Insert after a particular element \n 7. Delete from begin \n 8.Delete from end \n 9. Delete particular element \n 10 Search an element \n 11. Sort \n 12. Reverse \n 13. Count";

cin>>ch;

switch(ch)

{

case 1:

insertb();

break;

case 2:

display();

break;

case 3:

break;

case 4:

inserte();

break;

case 5:

insertbp();

break;

case 6:

insertap();

break;

case 7:

deleteb();

break;

case 8:

deletee();

break;

case 9:

deletep();

break;

case 10:

search();

break;

case 11:

sort();

break;

case 12:

reverse();

break;

case 13:

count();

break;

default:

cout<<endl<<"Invalid option";

}

}while(ch!=3);

}

void count()

{

if(list==NULL)

{

cout<<endl<<"list is empty :(";

}

else

{

q=list;

inti=0;

while(q->rptr!=NULL)

{

q=q->rptr;

i++;

}

cout<<endl<<"number of element "<<i+1;

}

}

void reverse()

{

if(list==NULL)

{

cout<<endl<<"List is empty";

}

else

{

struct node \*temp;

temp=NULL;

q=s=list;

r=q->rptr;

while(r!=NULL)

{

temp=q;

q=r;

r=q->rptr;

q->rptr=temp;

temp->lptr=q;

}

list=q;

s->rptr=NULL;

list->lptr=NULL;

}

}

void sort()

{

if(list==NULL)

{

cout<<endl<<"List is empty";

}

else

{

q=list;

while(q!=NULL)

{

r=q->rptr;

while(r!=NULL)

{

if(q->data > r->data)

{

int temp;

temp=q->data;

q->data=r->data;

r->data=temp;

}

r=r->rptr;

}

q=q->rptr;

}

}

}

void search()

{

if(list==NULL)

{

cout<<endl<<"List is empty";

}

else

{

inti=0;

q=list;

cout<<endl<<"Enter key value";

cin>>data1;

while(q->data!=data1 && q->rptr!=NULL)

{

i=i+1;

q=q->rptr;

}

if(q->data==data1)

{

cout<<endl<<"Data found at location "<<i;

}

else

{

cout<<endl<<"Data not found :( :(";

}

}

}

voiddeletep()

{

if(list==NULL)

{

cout<<endl<<"list is empty";

}

else

{

cout<<endl<<"Enter the element to be deleted";

cin>>data1;

q=list;

if(list->data==data1)

{

cout<<endl<<list->data<<" is deleted";

list=list->rptr;

free(q);

}

else

{

while(q->data!=data1 && q->rptr!=NULL)

{

r=q;

q=q->rptr;

}

if (q->rptr==NULL && q->data==data1)

{

cout<<q->data <<" is deleted";

r->rptr=NULL;

free(q);

}

else if(q->data==data1)

{

s=q->rptr;

cout<< q->data <<" is deleted";

r->rptr=s;

s->lptr=r;

free(q);

}

else

{

cout<<endl<<"Element mnot found :( :(";

}

}

}

}

voiddeletee()

{

if(list==NULL)

{

cout<<endl<<"list is empty";

}

else

{

q=list;

if(list->rptr==NULL)

{

cout<<endl<< list->data <<" is deleted";

free(list);

list=NULL;

}

else

{

while(q->rptr!=NULL)

{

r=q;

q=q->rptr;

}

r->rptr=NULL;

cout<<endl<< q->data <<" is deleted";

free(q);

}

}

}

voiddeleteb()

{

if(list==NULL)

{

cout<<endl<<"List is empty cant delete any element :( :(";

}

else

{

q=list;

list=list->rptr;

cout<<endl<< q->data <<" is deleted";

free(q);

}

}

voidinsertap()

{

if(list==NULL)

{

cout<<endl<<"List is empty";

}

else

{

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

cout<<endl<<"Enter element";

cin>>data1;

p->data=data1;

cout<<endl<<"Enter element after which you want to insert a new element";

cin>>data2;

q=list;

while(q->data!=data2 && q->rptr!=NULL)

{

q=q->rptr;

}

if(q->data==data2 && q->rptr!=NULL)

{

r=q->rptr;

q->rptr=p;

p->lptr=q;

p->rptr=r;

r->lptr=p;

}

else if(q->data==data2 && q->rptr==NULL)

{

q->rptr=p;

p->lptr=q;

p->rptr=NULL;

}

else

{

cout<<endl<<"Data not found";

}

}

}

voidinsertbp()

{

if(list==NULL)

{

cout<<endl<<"List is empty";

}

else

{

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

cout<<endl<<"Enter element";

cin>>data1;

p->data=data1;

cout<<endl<<"Enter element before which you want to insert a new element";

cin>>data2;

q=list;

if(q->data==data2)

{

p->lptr=NULL;

p->rptr=q;

q->lptr=p;

list=p;

}

else

{

while(q->data!=data2 && q->rptr!=NULL)

{

r=q;

q=q->rptr;

}

if(q->data==data2)

{

r->rptr=p;

p->lptr=r;

p->rptr=q;

q->lptr=p;

}

else

{

cout<<endl<<"Data not found";

}

}

}

}

voidinserte()

{

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

cout<<endl<<"Enter element";

cin>>data1;

p->data=data1;

if(list==NULL)

{

p->lptr=NULL;

p->rptr=NULL;

list=p;

}

else

{

q=list;

while(q->rptr!=NULL)

{

q=q->rptr;

}

q->rptr=p;

p->lptr=q;

p->rptr=NULL;

}

}

voidinsertb()

{

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

cout<<endl<<"Enter element";

cin>>data1;

p->data=data1;

if(list==NULL)

{

p->lptr=NULL;

p->rptr=NULL;

list=p;

}

else

{

p->lptr=NULL;

p->rptr=list;

list->lptr=p;

list=p;

cout<<"Value inserted";

}

}

void display()

{

if(list==NULL)

{

cout<<endl<<"List is empty";

}

else

{

q=list;

while(q!=NULL)

{

cout<< q->data <<"===>";

q=q->rptr;

}

}

}

};

int main()

{

doubly d;

d.menu();

return 0;

}

**Output:**

