/\*

A double-ended queue(deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a one-dimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque.

\*/

#include<iostream>

#include<process.h>

using namespace std;

#define size 5

struct queue

{

int que[size];

int front,rear;

}Q;

class DeQue

{

public:

DeQue();

int Qfull();

int Qempty();

int insert\_rear(int item);

int delete\_front();

int insert\_front(int item);

int delete\_rear();

void display();

};

DeQue::DeQue()

{

Q.front=-1;

Q.rear=-1;

for(int i=0;i<size;i++)

Q.que[i]=-1;

}

int DeQue::Qfull()

{

if(Q.rear==size-1)

return 1;

else

return 0;

}

int DeQue::Qempty()

{

if((Q.front>Q.rear)||(Q.front==-1&&Q.rear==-1))

return 1;

else

return 0;

}

int DeQue::insert\_rear(int item)

{

if(Q.front==-1 && Q.rear==-1)

Q.front++;

Q.que[++Q.rear]=item;

return Q.rear;

}

int DeQue::delete\_front()

{

int item;

if(Q.front==-1)

Q.front++;

item=Q.que[Q.front];

Q.que[Q.front]=-1;

Q.front++;

return item;

}

int DeQue::insert\_front(int item)

{

int i,j;

if(Q.front==-1)

Q.front++;

i=Q.front-1;

while(i>=0)

{

Q.que[i+1]=Q.que[i];

i--;

}

j=Q.rear;

while(j>=Q.front)

{

Q.que[j+1]=Q.que[j];

j--;

}

Q.rear++;

Q.que[Q.front]=item;

return Q.front;

}

int DeQue::delete\_rear()

{

int item;

item=Q.que[Q.rear];

Q.que[Q.rear]=-1; //logical deletion

Q.rear--;

return item;

}

void DeQue::display()

{

int i;

cout<<"\nStrength Queue is ::\n";

for(i=Q.front;i<=Q.rear;i++)

cout<<""<<Q.que[i];

}

int main()

{

int choice,item;

char ans;

ans='y';

DeQue obj;

cout<<"\n\t\t Program for simulation of doubly ended queues using arrays ::";

do

{

cout<<"\n1.Insert by rear\n2.Delete by front\n3.Insert by front\n4.Delete by rear";

cout<<"\n5.display\n6.Exit";

cin>>choice;

switch(choice)

{

case 1:

if(obj.Qfull())

cout<<"\nDoubly ended Queue is full";

else

{

cout<<"\nEnter the item to be inserted";

cin>>item;

Q.rear=obj.insert\_rear(item);

}

break;

case 2:

if(obj.Qempty())

cout<<"\nDoubly ended queue is Empty";

else

{

item=obj.delete\_front();

cout<<"\nThe item deleted from queue is"<<item;

}

break;

case 3:

if(obj.Qfull())

cout<<"\n Doubly link queued is full";

else

{

cout<<"\nEnter the item to be inserted::";

cin>>item;

Q.front=obj.insert\_front(item);

}

break;

case 4:

if(obj.Qempty())

cout<<"\nDoubly ended Queue is empty::";

else

{

item=obj.delete\_rear();

cout<<"\nThe item deleted from queue is"<<item;

}

break;

case 5:

obj.display();

break;

case 6:

exit(0);

}

cout<<"\nDo you want to continue?";

cin>>ans;

}while(ans=='y'||ans=='Y');

return 0;

}

/\*

OUTPUT :

Program for simulation of doubly ended queues using arrays ::

1.Insert by rear

2.Delete by front

3.Insert by front

4.Delete by rear

5.display

6.Exit1

Enter the item to be inserted 22

Do you want to continue?y

1.Insert by rear

2.Delete by front

3.Insert by front

4.Delete by rear

5.display

6.Exit3

Enter the item to be inserted::55

Do you want to continue?y

1.Insert by rear

2.Delete by front

3.Insert by front

4.Delete by rear

5.display

6.Exit5

Strength Queue is ::

5522

Do you want to continue?y

1.Insert by rear

2.Delete by front

3.Insert by front

4.Delete by rear

5.display

6.Exit2

The item deleted from queue is55

Do you want to continue?y

1.Insert by rear

2.Delete by front

3.Insert by front

4.Delete by rear

5.display

6.Exit4

The item deleted from queue is22

Do you want to continue?y

1.Insert by rear

2.Delete by front

3.Insert by front

4.Delete by rear

5.display

6.Exit5

Strength Queue is ::

Do you want to continue?

\*/