Practical 12

#include <iostream>

#include <stdio.h>

#define MAX 10

using namespace std;

struct que

{

int arr[MAX];

int front, rear;

};

void init(struct que \*q)

{

q->front = -1;

q->rear = -1;

}

void print(struct que q)

{

int i;

i = q.front;

while (i != q.rear)

{

cout << "\t" << q.arr[i];

i = (i + 1) % MAX;

}

cout << "\t" << q.arr[q.rear];

}

int isempty(struct que q)

{

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

return 1;

else

return 0;

}

int isfull(struct que q)

{

if ((q.rear + 1) % MAX == q.front)

return 1;

else

return 0;

}

void addf(struct que \*q, int data)

{

if (isempty(\*q))

{

q->front = q->rear = 0;

q->arr[q->front] = data;

}

else

{

q->front = (q->front - 1 + MAX) % MAX;

q->arr[q->front] = data;

}

}

void addr(struct que \*q, int data)

{

if (isempty(\*q))

{

q->front = q->rear = 0;

q->arr[q->rear] = data;

}

else

{

q->rear = (q->rear + 1) % MAX;

q->arr[q->rear] = data;

}

}

int delf(struct que \*q)

{

int data1;

data1 = q->arr[q->front];

if (q->front == q->rear)

init(q);

else

q->front = (q->front + 1) % MAX;

return data1;

}

int delr(struct que \*q)

{

int data1;

data1 = q->arr[q->rear];

if (q->front == q->rear)

init(q);

else

q->rear = (q->rear - 1 + MAX) % MAX;

return data1;

}

int main()

{

struct que q;

int data, ch;

init(&q);

while (ch != 6)

{

cout << "\t\n1.Insert front,"

"\t\n2.Insert rear,"

"\t\n3.Delete front,"

"\t\n4.Delete rear,"

"\t\n5.Print,"

"\t\n6.Exit";

cout << "\n\tEnter your choice";

cin >> ch;

switch (ch)

{

case 1:

cout << "\nEnter data to insert front: ";

cin >> data;

addf(&q, data);

break;

case 2:

cout << "\nEnter the data to insert rear: ";

cin >> data;

addr(&q, data);

break;

case 3:

if (isempty(q))

cout << "\nDequeue is empty";

else

{

data = delf(&q);

cout << "\nDeleted data is: " << data;

}

break;

case 4:

if (isempty(q))

cout << "\nDequeue is empty";

else

{

data = delr(&q);

cout << "\nDeleted data is" << data;

}

break;

case 5:

if (isempty(q))

cout << "\nDequeue is empty";

else

{

cout << "\nDequeue elements are:";

print(q);

}

break;

}

}

return 0;

}

