**/\*WAP for implementation of Circular Queue\*/**

**#include<iostream>**

**using namespace std;**

**class queue**

**{**

**int FRONT;**

**int REAR;**

**int MAX;**

**int \*arr;**

**public:**

**queue()**

**{**

**cout<<"Enter the size of queue:\t";**

**cin>>MAX;**

**FRONT=-1;**

**REAR=-1;**

**arr = new int[MAX];**

**}**

**bool is\_empty()**

**{**

**if (FRONT==-1)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**bool is\_full()**

**{**

**if (((REAR==MAX-1)&& (FRONT==0))|| (FRONT == REAR+1))**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**void enqueue(int num)**

**{**

**if (REAR ==-1 && FRONT==-1)**

**{**

**FRONT=0;**

**REAR=0;**

**arr[REAR]=num;**

**}**

**else if (!is\_full())**

**{**

**if(REAR==MAX-1)**

**{**

**REAR=0;**

**}**

**else**

**{**

**REAR+=1;**

**}**

**arr[REAR]=num;**

**}**

**else**

**{**

**cout<<num<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Overflow\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;**

**}**

**}**

**int dequeue()**

**{**

**if(!is\_empty())**

**{**

**int VALUE=arr[FRONT];**

**if (FRONT==REAR)**

**{**

**FRONT=-1;**

**REAR=-1;**

**}**

**else if (FRONT==MAX-1)**

**{**

**FRONT=0;**

**}**

**else**

**{**

**FRONT+=1;**

**}**

**return VALUE;**

**}**

**}**

**};**

**int main()**

**{**

**queue q;**

**int choice;**

**int num;**

**do**

**{**

**cout<<endl<<"Enter your choice:"<<endl;**

**cout<<"1)enqueue"<<endl;**

**cout<<"2)dequeue"<<endl;**

**cout<<"3)quit"<<endl;**

**cout<<"Enter your choice:\t";**

**cin>>choice;**

**switch(choice)**

**{**

**case 1:**

**cout<<"\nEnter number to enqueue:\t";**

**cin>>num;**

**q.enqueue(num);**

**break;**

**case 2:**

**if (!q.is\_empty())**

**{**

**cout<<"\n\*\*\*\*\*\*\*\*\*Dequeued number:"<<q.dequeue()<<"\*\*\*\*\*\*\*\*\*\n";**

**}**

**else**

**{**

**cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Underflow\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;**

**}**

**break;**

**case 3:**

**break;**

**}**

**}**

**while (choice !=3);**

**}**

**/\*WAP for implementation of Circular Queue\*/**

**#include <iostream>**

**#define MAX 4**

**using namespace std;**

**int front=-1,rear=-1,data,queue[MAX];**

**void push()**

**{**

**if ((front == 0 && rear == MAX-1) || (front == rear+1))**

**{**

**cout<<"Queue Overflow \n";**

**return;**

**}**

**if (front == -1 && rear==-1)**

**{**

**front = 0;**

**rear = 0;**

**}**

**else**

**{**

**if (rear == MAX - 1)**

**rear = 0;**

**else**

**rear = rear + 1;**

**}**

**cout<<"Enter value to push: ";**

**cin>>data;**

**queue[rear] = data ;**

**}**

**void pop()**

**{**

**if (front == -1)**

**{**

**cout<<"Queue Underflow\n";**

**return ;**

**}**

**cout<<"Element dequeued from Circural queue is : "<<queue[front]<<endl;**

**if (front == rear)**

**{**

**front = -1;**

**rear = -1;**

**}**

**else**

**{**

**if (front == MAX - 1)**

**front = 0;**

**else**

**front = front + 1;**

**}**

**}**

**void display()**

**{**

**{**

**int f = front, r = rear;**

**if (front == -1)**

**{**

**cout<<"Queue is empty"<<endl;**

**return;**

**}**

**cout<<"Queue elements are :\n";**

**if (f <= r)**

**{**

**while (f <= r)**

**{**

**cout<<queue[f]<<" ";**

**f++;**

**}**

**}**

**else**

**{**

**while (f <= MAX - 1)**

**{**

**cout<<queue[f]<<" ";**

**f++;**

**}**

**f = 0;**

**while (f <= r)**

**{**

**cout<<queue[f]<<" ";**

**f++;**

**}**

**}**

**cout<<endl;**

**}**

**}**

**int main()**

**{**

**int ch;**

**cout << "1) Insert element to circular queue" << endl;**

**cout << "2) Delete element from circular queue" << endl;**

**cout << "3) Display all the elements of ciruclar queue" << endl;**

**cout << "4) Exit" << endl;**

**do**

**{**

**cout << "Enter your choice: ";**

**cin >> ch;**

**cout<<endl;**

**switch (ch)**

**{**

**case 1:**

**push();**

**break;**

**case 2:**

**pop();**

**break;**

**case 3:**

**display();**

**break;**

**case 4:**

**cout << "Exit" << endl;**

**break;**

**default:**

**cout << "Invalid choice" << endl;**

**}**

**cout<<endl<<endl<<endl;**

**}**

**while (ch != 4);**

**return 0;**

**}**

**/\*WAP for implementation of Circular Queue\*/**

**#include<iostream>**

**#include<cstdlib>**

**#define capacity 4**

**using namespace std;**

**template <class X>**

**class Queue**

**{**

**int front1,rear;**

**X data[capacity];**

**public:**

**Queue():front1(-1),rear(-1) {}**

**void enqueue(X var)**

**{**

**if(IsQueuefull())**

**cout<<"queue overflow"<<endl;**

**else**

**{**

**rear=(rear+1)%capacity;**

**data[rear]=var;**

**if(front1 == -1)**

**front1 =0;**

**}**

**}**

**void dequeue()**

**{**

**if(IsEmptyQueue())**

**cout<<"queue underflow"<<endl;**

**else**

**{**

**cout<<"\ndequeued element is "<<data[front1]<<endl;**

**if(front1==rear)**

**front1=rear=-1;**

**else**

**front1=(front1+1)%capacity;**

**}**

**}**

**bool IsEmptyQueue()**

**{**

**if(front1 == -1)**

**return true;**

**else**

**return false;**

**}**

**bool IsQueuefull()**

**{**

**if((front1==0&&rear==capacity-1)||front1==rear+1)**

**return true;**

**else**

**return false;**

**}**

**int Queuesize()**

**{**

**if(IsEmptyQueue())**

**return 0;**

**else**

**return ((capacity-front1+rear)%capacity+1);**

**}**

**void Front()**

**{**

**if(IsEmptyQueue())**

**cout<<"\nQueue underflow"<<endl;**

**else**

**cout<<"\nThe front element of queue is : "<<data[front1]<<endl;**

**}**

**};**

**int main()**

**{**

**Queue <int> q;**

**int choice;**

**while(1)**

**{**

**cout<<"1.Enqueue\n2.Dequeue\n3.View front element\n4.Check queue size\n5.exit\nEnter your choice ";**

**cin>>choice;**

**switch(choice)**

**{**

**case 1:**

**{**

**while(1)**

**{**

**int num;**

**cout<<"\nEnter -1 to finish enqueue\nEnter the value: ";**

**cin>>num;**

**if(num==-1)**

**break;**

**q.enqueue(num);**

**}**

**break;**

**}**

**case 2:**

**{**

**q.dequeue();**

**break;**

**}**

**case 3:**

**{**

**q.Front();**

**break;**

**}**

**case 4:**

**{**

**cout<<"\nQueue size is: "<<q.Queuesize();**

**break;**

**}**

**default:**

**exit(0);**

**}**

**cout<<"\n\n";**

**}**

**return 0;**

**}**

**/\*WAP for implementation of Circular Queue\*/**

**#include<iostream>**

**#define max 5**

**using namespace std;**

**//define a Queue //**

**template<class T>**

**class Queue**

**{**

**private:**

**int front,rear,counter;**

**T arr[max];**

**T sign;**

**public:**

**// constructor to initialize front and rear**

**Queue(T emptysign)**

**{**

**front=-1;**

**rear=-1;**

**counter=0 ;**

**sign=emptysign;**

**for(int i= 0; i<max; i++)**

**{**

**arr[i]=sign;**

**}**

**}**

**//isEmpty to check if queue is empty**

**bool isEmpty()**

**{**

**if ( counter == 0)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**//to check if Queue is full**

**bool isFull()**

**{**

**if (counter == max)**

**{**

**return true;**

**}**

**else**

**{**

**return false;**

**}**

**}**

**//enqueue into Queue**

**void enq(T data)**

**{**

**if(!isFull())**

**{**

**if(front == -1)**

**front = 0;**

**arr[++rear % max] = data;**

**counter = rear - front + 1;**

**}**

**else**

**{**

**cout<<"Overflow"<<endl;**

**}**

**}**

**//dequeue from the Queue**

**void deq()**

**{**

**if(!isEmpty())**

**{**

**cout << arr[front % max] << endl;**

**arr[front++ % max]=sign;**

**counter = rear - front + 1;**

**}**

**else**

**{**

**cout<<"UnderFlow"<<endl;**

**}**

**}**

**//display Queue**

**void display()**

**{**

**cout<<"\n ==========="<<endl;**

**cout<<"The queue is ==>\t";**

**for(int i=0; i<max; i++)**

**{**

**cout<<arr[i]<<"\t";**

**}**

**cout<<"front:: "<<front%max<<"\tlen:: "<<counter<<endl;**

**cout<<" ===========\n"<<endl;**

**}**

**};**

**//driver main function**

**int main()**

**{**

**Queue<int> que(0);**

**char opt='a';**

**int val;**

**cout<<"what to do:\n"<<"d for dequeue:\n"<<"e for enqueue\n"<<"x for display\n"<<"n for end"<<endl;**

**while(opt!='n')**

**{**

**cout<<"your choice: ";**

**cin>>opt;**

**switch(opt)**

**{**

**case 'd':**

**que.deq();**

**break;**

**case 'e':**

**cout<<"enter value:";**

**cin >> val;**

**que.enq(val);**

**break;**

**case 'x':**

**que.display();**

**break;**

**case 'n':**

**cout<<"thank you"<<endl;**

**break;**

**}**

**}**

**return 0;**

**}**