AIM: - Queues using array

Algorithm:-

Insert

Step 1: If maxsize-1=rear then this means the queue is full.

Step 2: But if rear<maxsize means that we can store an element in an array.

Step 3: So increment the rear value by 1 and then insert an element at the rear index.

Delete

Step 4: If front=-1 or front>rear then no element is available to delete.

Step 5: else delete front index element.

Step 6: if rear=front then set-1 to both front and rear.

Step 7: else we increment front.

Display

Step 8: first check whether the queue is not empty.

Step 9: if empty we display that the queue is empty we simply return from the function and not execute further inside the function.

Step 10: else print all elements from front to rear.

Program: -

#include<stdio.h>//standard input output header file//

#include<stdlib.h>//standard library header file//

#define maxsize 10//defining size//

void insert();//declaring insert function//

void delete();//declaring delete function//

void display();//declaring display function//

int queue[maxsize]; //declaring queue maxsize//

int front=-1,rear=-1;//initializing the values of front and the rear//

int main()//main function//

{

int choice;//declaring the choice//

while(1)//while loop//

{

printf("\*\*\*\*MENU\*\*\*\*\n");//prints the statement//

printf("------------\n");//prints the statement//

printf("\n 1.insert \n 2.delete \n 3.display \n 4.exit \n");//prints the statement//

printf("enter the operation number to be performed::\n");//prints the statement//

scanf("%d",&choice);//scans the choice and declares the memory location//

switch(choice)//switch case//

{

case 1:

insert();//insert function//

break;

case 2:

delete();//delete function//

break;

case 3:

display();//display function//

break;

case 4:

exit(0);

default:printf("invalid entry\n");//prints the statement//

}

}

return 0;

}

void insert()//insert function//

{

int item;//declaring the item//

printf("enter the element to be inserted::\n");//prints the statement//

scanf("%d",&item);//scans the statement and declares the memory location//

if(rear==maxsize-1)//checks the condition//

{

printf("overflow\n");//prints the statement//

}

if(front==-1&&rear==-1)//checks the condition//

{

front=0;

rear=0;

}

else

{

rear=rear+1;//rare value is incremented to the next value//

}

queue[rear]=item;//initializes the rare value of queue to item//

printf("value inserted\n");//prints the statement//

}

void delete()//delete function//

{

int item;//initializes the item//

if(front==-1||front>rear)//checks the condition//

{

printf("underflow\n");//prints the statement//

}

else

{

item=queue[front];//item is to the initializes the queue front//

if(front==rear)//checks the condition//

{

front=-1;

rear=-1;

}

else

{

front=front+1;//front value is initialized to the next value//

}

printf("value deleted\n");//prints the statement//

}

}

void display()//display function//

{

int i;//initializes the i//

if(rear==-1)//checks the condition//

{

printf("queue is empty\n");//prints the statement//

}

else

{

printf("elements present in the queues are::\n");//prints the statement//

for(i=front;i<=rear;i++)//enters into the loop//

{

printf("\n%d\n",queue[i]); //prints the statement//

}

}

}

Screenshot of the output:-

