|  |  |
| --- | --- |
| **Ex.No.2** | **QUEUE OPERATIONS** |
| **DATE:03.01.2025** |

**AIM:**

To implement a Queue data structure and perform the basic operations on the queue.

**ALGORITHM:**

STEP 1**:** Start the program

STEP 2: Declare an array queue with the required size.

STEP 3: Initialize rear and front variables to -1

STEP 4: Create a Menu choice for Insert, delete and display operations on the Queue

STEP 5: insert();

1. Check queue overflow.
2. If not full, Get the input value for the item to be added
3. Increment the rear value
4. Place the item at the rear position

STEP 6: delete();

1. Check queue underflow.
2. If not empty, then display the item at front position
3. Increment front value by 1

STEP 7: display(): Used to display the items in the queue

STEP 8: Stop the execution

**PROGRAM CODING:**

#include <stdio.h>

#include<stdlib.h>

#define MAX 5

void insert();

void delete();

void display();

int queue[MAX];

int rear = - 1;

int front = - 1;

int main()

{

int choice;

while (1)

{

printf("1.Insert element to queue \n");

printf("2.Delete element from queue \n");

printf("3.Display all elements of queue \n");

printf("4.Quit \n");

printf("Enter your choice : ");

scanf("%d", &choice);

switch (choice)

{

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

exit(1);

default:

printf("Wrong choice \n");

} /\* End of switch \*/

} /\* End of while \*/

return 0;

} /\* End of main() \*/

void insert()

{

int item;

if (rear == MAX - 1)

printf("Queue Overflow \n");

else

{

if (front == - 1)

/\*If queue is initially empty \*/

front = 0;

printf("Inset the element in queue : ");

scanf("%d", &item);

rear = rear + 1;

queue[rear] = item;

}

} /\* End of insert() \*/

void delete()

{

if (front == - 1 || front > rear)

{

printf("Queue Underflow \n");

return ;

}

else

{

printf("Element deleted from queue is : %d\n", queue[front]);

front = front + 1;

}

} /\* End of delete() \*/

void display()

{

int i;

if (front == - 1)

printf("Queue is empty \n");

else

{

printf("Queue is : \n");

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

printf("%d ", queue[i]);

printf("\n");

}

} /\* End of display() \*/

**RESULT:**

Thus the above program executed successfully and queue operations are implemented.