Experiment No.4
Implementation of Queue menu driven program using arrays
Name : Saurabh Mane
Roll No: 25
Date of Performance: 8/8/23
Date of Submission:22/8/23
Marks:
Sign:

Experiment No. 4: Simple Queue Operations

Aim: To implement a Linear Queue using arrays.

Objective:

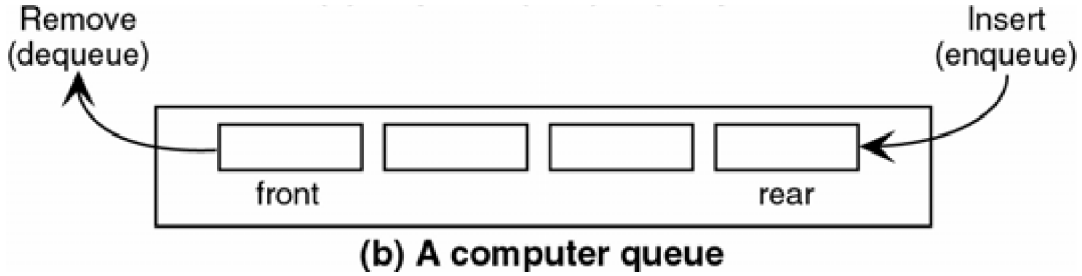
1 Understand the Queue data structure and its basic operations.

2. Understand the method of defining Queue ADT and its basic operations.

3. Learn how to create objects from an ADT and member functions are invoked.

Theory:

A queue is an ordered collection where items are removed from the front and inserted at the rear, following the First-In-First-Out (FIFO) order. The fundamental operations for a queue are "Enqueue," which adds an item to the rear, and "Dequeue," which removes an item from the front.



(b) A computer queue

Typically, a one-dimensional array is used to implement a queue, and two integer values, FRONT and REAR, track the front and rear positions in the array. When an element is removed from the queue, FRONT is incremented by one, and when an element is added to

the queue, REAR is increased by one. This ensures that items are processed in the order they were added, maintaining the FIFO principle.

Algorithm:

ENQUEUE(item)

1. If (queue is full)

   Print "overflow"

2. if (First node insertion)

   Front++

3. rear++

Queue[rear]=value

DEQUEUE()

1. If (queue is empty)

   Print "underflow"

2. if(front=rear)

   Front=-1 and rear=-1

3. t = queue[front]
4. front++
5. Return t

ISEMPTY()

1. If(front = -1)then

   return 1

2. return 0

ISFULL()

1. If(rear = max)then

   return 1

2. return 0

Code:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
// Changing this value will change length of array
int queue[MAX];
int front = -1, rear = -1;
void insert(void);
int delete_element(void);
int peek(void);
void display(void);
int main()
{ int option, val;
do {
printf("\n\n ***** MAIN MENU *****");
printf("\n 1. Insert an element");
printf("\n 2. Delete an element");
printf("\n 3. Peek");
printf("\n 4. Display the queue");
printf("\n 5. EXIT");
printf("\n Enter your option : ");
scanf("%d", &option);
switch(option)
{
```

```
 case 1:
 insert();
 break;
case 2:
 val = delete_element();
 if (val != -1)
printf("\n The number deleted is : %d", val);
 break;
 case 3:
 val = peek(); if (val != -1)
 printf("\n The first value in queue is : %d", val);
break;
 case 4:
 display();
 break;
 }
 }
while(option != 5);
getch();
return 0;
}
void insert() {
 int num;
printf("\n Enter the number to be inserted in the queue : ");
scanf("%d", &num);
 if(rear == MAX-1)
```

```
printf("\n OVERFLOW");
else if(front == -1 && rear == -1)
front = rear = 0;
else
rear++;
 queue[rear] = num;
}
 int delete_element() {
 int val;
 if(front == -1 || front>rear) {
printf("\n UNDERFLOW");
return -1;
 } else {
val = queue[front];
 front++;
if(front > rear)
front = rear = -1;
 return val;
 }
 }
int peek() {
if(front==-1 || front>rear) {
 printf("\n QUEUE IS EMPTY");
return -1;
 }
 else {
```

```
return queue[front]; }
 }
void display() {
 int i;
printf("\n");
 if(front == -1 || front > rear)
 printf("\n QUEUE IS EMPTY");
else {
 for(i = front;i <= rear;i++)
printf("\t %d", queue[i]);
 }
 }
```

Output:

```
***** MAIN MENU *****"
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT
Enter your option : 1
Enter the number to be inserted in the queue : 50
```

Conclusion:

What is the structure of queue ADT?

The Queue Abstract Data Type (ADT) is a linear data structure that follows the First-In-First-Out (FIFO) principle, meaning that the first element added to the queue is the first one to be removed. The basic operations and characteristics of a queue ADT include:

1. Enqueue:Adds an element to the back (rear) of the queue.

2. Dequeue: Removes and returns the element from the front (head) of the queue.

3. Peek (or Front): Allows you to view the element at the front of the queue without removing it.

4. IsEmpty: Checks if the queue is empty.

5. Size: Returns the number of elements currently in the queue.

A simple real-life analogy for a queue is a line of people waiting for a service, where the person who arrives first is the first to be served.

List various applications of queues?

Here are various applications of queues:

1. Print Queue: Managing print jobs in order.

2. Task Scheduling: Scheduling tasks in operating systems.

3. Breadth-First Search: Traversing graphs level by level.

4. Call Center Systems: Handling customer service calls.

5. Buffer Management: Storing data in a temporary buffer.

6. Request Handling: Managing requests in web servers.

7. Bounded Buffer: Handling data between producers and consumers.

8. Job Scheduling: Scheduling tasks in computing clusters.

9. Simulation Systems: Modeling real-world scenarios.

10. Task Queues: Managing background tasks in applications.

Where is queue used in a computer system proceesing?

Queues are used in a computer system for tasks like:

1. Task scheduling in operating systems.
2. Managing I/O operations.
3. Handling print jobs.
4. Efficient interrupt handling.
5. Buffer management.
6. Job queues in computing clusters.
7. Synchronization.
8. Message queues in distributed systems.
9. Request handling in web servers.
10. Background task management in applications.

They ensure orderly and efficient processing of tasks, data, and requests.