DATA STRUCTURES LABORATORY

Course Code: 1BCSL306

VTU 2025 Scheme – Semester 3

EXPERIMENT – 04

Printer Queue Simulation

Queue Implementation using Array

Simple Code • Easy to Remember • Easy to Write in Exam

Your VTU Study Partner

Experiment 04

1. Aim

Aim

To develop a menu-driven C program to simulate a Printer Queue using a queue data structure.

2. VTU Prescribed Experiment

VTU Prescribed Experiment

Develop a menu-driven C program to simulate a **Printer Queue** using a Queue data structure.
The program shall support the following operations:

a. Add a print job to the printer queue.

b. Process/Delete a print job from the printer queue.

c. Display the number of print jobs waiting in the queue.

d. Demonstrate Queue Overflow and Queue Underflow situations.

e. Exit.

Use appropriate functions to perform the above operations.

3. Simple Theory

Queue:

A queue is a linear data structure in which insertion is performed at the **REAR** and deletion is performed from the **FRONT**.

Queue follows:

$$\boxed{FIFO}$$

where:

$$\boxed{FIFO = First\ In\ First\ Out}$$

In a printer queue, the print job that enters first is printed first.

4. Printer Queue Example

Suppose three print jobs are added:

$$101, \quad 102, \quad 103$$

Then:

$$\boxed{101 \rightarrow 102 \rightarrow 103}$$

Here:

$$\boxed{FRONT = 101}$$

and:

$$\boxed{REAR = 103}$$

The first job processed will be:

$$\boxed{101}$$

5. Basic Queue Operations

5.1. Enqueue

Enqueue means adding a print job at the rear end.

$$\boxed{rear = rear + 1}$$

Then:

$$\boxed{queue[rear] = job}$$

5.2. Dequeue

Dequeue means processing/removing the print job from the front.
The element at:

$$\boxed{queue[front]}$$

is processed.
Then:

$$\boxed{front = front + 1}$$

5.3. Overflow

Overflow occurs when a new print job is added to a full queue.
Condition:

$$\boxed{rear == MAX - 1}$$

5.4. Underflow

Underflow occurs when we try to process a print job from an empty queue.
Condition:

$$\boxed{front == -1}$$

or when:

$$\boxed{front > rear}$$

6. Algorithm

6.1. Main Algorithm

1. Start.
2. Declare an array to represent Printer Queue.
3. Initialize:

$$front = -1, \qquad rear = -1$$

4. Display the menu:
   - Add Print Job
   - Process Print Job
   - Display Number of Jobs
   - Exit
5. Read user's choice.
6. Call the required function.
7. Repeat until Exit is selected.
8. Stop.

6.2. Algorithm for Add Print Job

1. Check whether:

$$rear == MAX - 1$$

2. If true, display Queue Overflow.
3. Otherwise read the Print Job ID.
4. If it is the first job:

$$front = 0$$

5. Increment rear.
6. Insert the job at rear.

6.3. Algorithm for Process Print Job

1. Check whether queue is empty.
2. If empty, display Queue Underflow.
3. Otherwise display the job at FRONT.
4. Increment FRONT.
5. If all jobs are processed, reset:

$$front = rear = -1$$

6.4. Algorithm to Display Number of Jobs

1. Check whether queue is empty.
2. If empty:

$$Jobs = 0$$

3. Otherwise calculate:

$$\boxed{Jobs = rear - front + 1}$$

4. Display the number of jobs waiting.

Easy Exam Memory

Remember:

$$\boxed{FIFO}$$

Add Job:

$$\boxed{Insert\ at\ REAR}$$

Process Job:

$$\boxed{Delete\ from\ FRONT}$$

Overflow:

$$\boxed{rear == MAX - 1}$$

Underflow:

$$\boxed{front == -1}$$

Number of Jobs:

$$\boxed{rear - front + 1}$$

7. C Program – Easy Student Version

```c
#include<stdio.h>

#define MAX 20

int q[MAX];
int front=-1, rear=-1;

void addJob()
{
    int job;

    if(rear==MAX-1)
    {
        printf("Queue Overflow\n");
        return;
    }

    printf("Enter Print Job ID: ");
    scanf("%d",&job);

    if(front==-1)
        front=0;

    q[++rear]=job;

    printf("Print Job Added\n");
}

void processJob()
{
    if(front==-1)
    {
        printf("Queue Underflow\n");
        return;
    }

    printf("Processed Job = %d\n",q[front]);

    front++;

    if(front>rear)
        front=rear=-1;
}

void displayCount()
{
    if(front==-1)
        printf("Jobs Waiting = 0\n");
    else
        printf("Jobs Waiting = %d\n",rear-front+1);
```

```
}

int main()
{
    int ch;

    do
    {
        printf("\n1.Add Print Job");
        printf("\n2.Process Print Job");
        printf("\n3.Display Jobs Waiting");
        printf("\n4.Exit");

        printf("\nEnter Choice: ");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1:
                addJob();
                break;

            case 2:
                processJob();
                break;

            case 3:
                displayCount();
                break;

            case 4:
                printf("Program Ended\n");
                break;

            default:
                printf("Invalid Choice\n");
        }

    }while(ch!=4);

    return 0;
}
```


8. Simple Program Explanation

8.1. Queue Declaration

```
#define MAX 20

int q[MAX];
```

```
int front=-1, rear=-1;
```

The queue can store a maximum of 20 print jobs.

Initially:

$$front = -1$$

and:

$$rear = -1$$

which means the queue is empty.

8.2. addJob() Function

First:

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

checks Queue Overflow.

If the first job is inserted:

```
if(front==-1)
    front=0;
```

Then the job is inserted using:

```
q[++rear]=job;
```

8.3. processJob() Function

If:

```
front==-1
```

the queue is empty.

Otherwise:

```
printf("%d",q[front]);
front++;
```

processes the first print job.

When all jobs are completed:

```
if(front>rear)
    front=rear=-1;
```

The queue becomes empty again.

8.4. displayCount() Function

The number of waiting jobs is calculated as:

$$\boxed{rear - front + 1}$$

If the queue is empty:

$$\boxed{Jobs = 0}$$

9. Queue Working Example

After adding:

$$101, \quad 102, \quad 103$$

the queue is:

$$\boxed{FRONT \rightarrow 101 \rightarrow 102 \rightarrow 103 \leftarrow REAR}$$

Number of jobs:

$$rear - front + 1$$

$$= 2 - 0 + 1$$

$$\boxed{= 3}$$

After processing one job:

$$101$$

is removed.

Remaining queue:

$$\boxed{102 \rightarrow 103}$$

Number of jobs waiting:

$$\boxed{2}$$

Easy Exam Memory

The complete program is only:

$$\boxed{Queue \rightarrow Front/Rear \rightarrow Add \rightarrow Process \rightarrow Count \rightarrow Main}$$

Add:

$$\boxed{Check\ Full \rightarrow Read\ Job \rightarrow Set\ Front \rightarrow ++Rear \rightarrow Insert}$$

Process:

$$\boxed{Check\ Empty \rightarrow Print\ Front \rightarrow Front++}$$

Count:

$$\boxed{rear - front + 1}$$

10. Sample Input and Output

```
1.Add Print Job
2.Process Print Job
3.Display Jobs Waiting
4.Exit

Enter Choice: 1
Enter Print Job ID: 101
Print Job Added

Enter Choice: 1
Enter Print Job ID: 102
Print Job Added

Enter Choice: 1
Enter Print Job ID: 103
Print Job Added


Enter Choice: 3
Jobs Waiting = 3


Enter Choice: 2
Processed Job = 101


Enter Choice: 3
Jobs Waiting = 2


Enter Choice: 2
Processed Job = 102

Enter Choice: 2
Processed Job = 103


Enter Choice: 3
Jobs Waiting = 0


Enter Choice: 2
Queue Underflow


Enter Choice: 4
Program Ended
```

11. Overflow Demonstration

Maximum queue size is:

$$MAX = 20$$

When:

$$rear = 19$$

the queue is full.

If another print job is added:

$$rear == MAX - 1$$

Therefore:

```
Queue Overflow
```

is displayed.

12. Underflow Demonstration

When:

$$front = -1$$

the printer queue is empty.

If Process Job is selected:

```
Queue Underflow
```

is displayed.

13. Simple Test Cases

Test	Operation	Expected Result
1	Add Job 101	Job 101 should be inserted.
2	Add Jobs 102 and 103	Three jobs should be waiting.
3	Process one job	Job 101 should be processed first.
4	Display jobs waiting	Two jobs should be displayed.
5	Process from empty queue	Queue Underflow should be displayed.
6	Add when queue is full	Queue Overflow should be displayed.

14. Result

Result

Thus, the C program to simulate a Printer Queue using a Queue data structure was successfully implemented.
The program performs:

- Add Print Job
- Process/Delete Print Job
- Display Number of Jobs Waiting
- Queue Overflow Demonstration
- Queue Underflow Demonstration

15. Important Viva Questions

Important Viva Questions

1. What is a Queue?
A queue is a linear data structure in which insertion takes place at REAR and deletion takes place from FRONT.

2. Which principle does Queue follow?

$$\boxed{FIFO}$$

First In First Out.

3. What is Enqueue?
Enqueue is the operation of inserting an element into the queue.

4. What is Dequeue?
Dequeue is the operation of deleting an element from the queue.

5. Where is insertion performed?
At the REAR.

6. Where is deletion performed?
At the FRONT.

7. What is Queue Overflow?
Overflow occurs when an element is inserted into a full queue.

8. What is the overflow condition?

$$\boxed{rear == MAX - 1}$$

9. What is Queue Underflow?
Underflow occurs when deletion is attempted from an empty queue.

10. What is the initial value of FRONT and REAR?

$$\boxed{front = -1, \qquad rear = -1}$$

11. How do we calculate the number of jobs waiting?

$$\boxed{rear - front + 1}$$

12. Why is Queue suitable for printer scheduling?
Because the print job that arrives first should normally be processed first.

13. What is the time complexity of Enqueue?

$$\boxed{O(1)}$$

14. What is the time complexity of Dequeue?

$$\boxed{O(1)}$$

15. Give one real-life example of Queue.
Printer queue, ticket counter, or customer service queue.

16. 1-Minute Exam Revision

Easy Exam Memory

Principle

$$\boxed{FIFO}$$

Initial Values

$$\boxed{front = -1, \quad rear = -1}$$

Insertion

$$\boxed{q[++rear] = job}$$

Deletion

$$\boxed{q[front]}$$

then:

$$\boxed{front++}$$

Overflow

$$\boxed{rear == MAX - 1}$$

Underflow

$$\boxed{front == -1}$$

Number of Waiting Jobs

$$\boxed{rear - front + 1}$$

17. Easy Code Memory Map

Remember

Remember:

$$\boxed{MAX \rightarrow Queue[] \rightarrow Front \rightarrow Rear}$$

Then only three functions:

$$\boxed{addJob()}$$

$$\boxed{processJob()}$$

$$\boxed{displayCount()}$$

Finally:

$$\boxed{main() \rightarrow menu \rightarrow switch}$$

Easy memory:

$$\boxed{Add = REAR}$$

$$\boxed{Process = FRONT}$$

$$\boxed{Queue = FIFO}$$

SEARCH CREATORS

1BCSL306 – Data Structures Laboratory

EXPERIMENT 04
COMPLETED

Printer Queue Simulation

Queue Implementation using Array

Understand → Remember → Write → Execute → Viva

Your VTU Study Partner