**K. J. Somaiya College of Engineering, Mumbai**
(A Constituent College of Somaiya Vidyavihar University)
**Department of Computer Engineering**

| |
|---|
| **Batch: A2 Roll No.: 16010122041** |
| **Experiment / assignment / tutorial No.** |
| **Grade: AA / AB / BB / BC / CC / CD /DD** |
| **Signature of the Staff In-charge with date** |

**Title:** Implementation of Basic operations on queue for the assigned application using Array and Linked List- Create, Insert, Delete, Destroy

**Objective:** To implement Basic Operations on Queue i.e. Create, Push, Pop, Display for the given application

**Expected Outcome of Experiment:**

| CO | Outcome |
|---|---|
| 1 | Explain the different data structures used in problem solving |

**Books/ Journals/ Websites referred:**


1. *Fundamentals Of Data Structures In C* – Ellis Horowitz, Satraj Sahni, Susan Anderson-Fred
2. *An Introduction to data structures with applications* – Jean Paul Tremblay, Paul G. Sorenson
3. *Data Structures A Pseudo Approach with C* – Richard F. Gilberg & Behrouz A. Forouzan
4.

**Abstract**:
(Define Queue, enlist queue operations).

Queue- A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First In First Out (FIFO). A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added

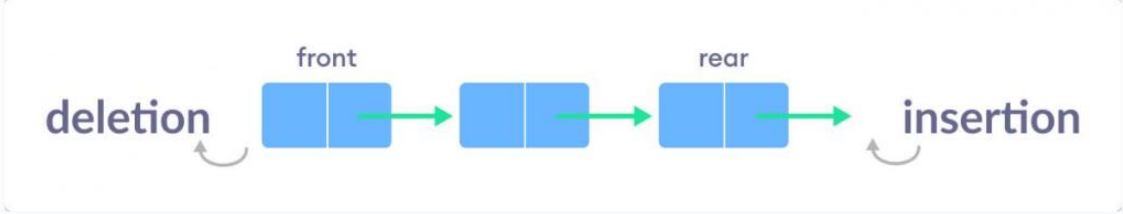
**List 5 Real Life applications of Queue:**

- A queue of people at ticket-window.
- Vehicles on toll-tax bridge.
- Phone answering system.
- Luggage checking machine.
- Patients waiting outside the doctor's clinic.

**Define and explain various types of queue with suitable diagram and their application(s):**

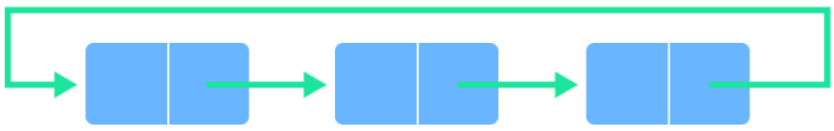
**1.Simple Queue**

In a simple queue, insertion takes place at the rear and removal occurs at the front. It strictly follows the FIFO (First in First out) rule.




**2.Circular Queue**

In a circular queue, the last element points to the first element making a circular link.

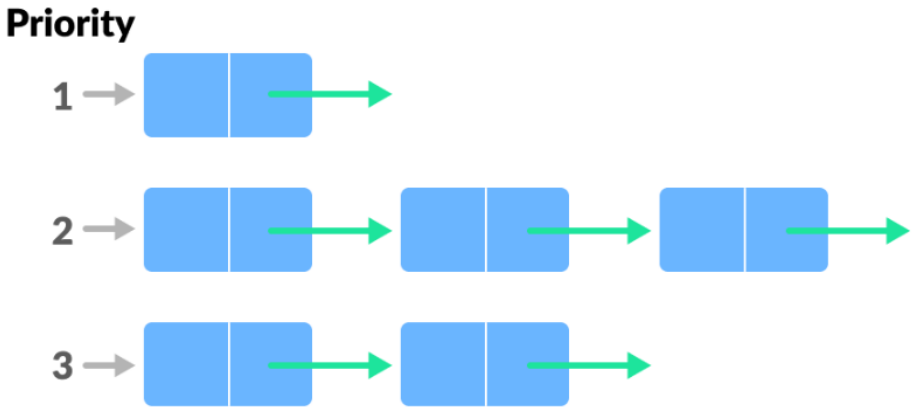


Circular Queue Representation

The main advantage of a circular queue over a simple queue is better memory utilization. If the last position is full and the first position is empty, we can insert an element in the first position. This action is not possible in a simple queue.

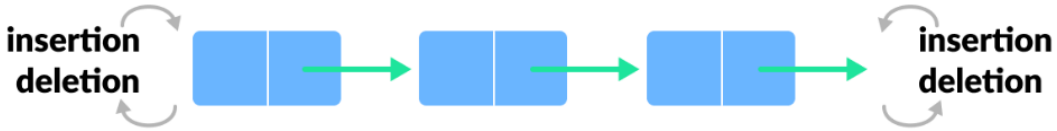
**3.Priority Queue**

A priority queue is a special type of queue in which each element is associated with a priority and is served according to its priority. If elements with the same priority occur, they are served according to their order in the queue.




**4.Deque (Double Ended Queue)**

In a double ended queue, insertion and removal of elements can be performed from either from the front or rear. Thus, it does not follow
the FIFO (First In First Out) rule.




**Queue ADT:**

**Value definition**

Abstract typedef <front, size> Queue
Condition: size != 0 && front = 0 && rear = -1;

**Operator definition**

Abstract Queue isempty()
Post Condition : true if rear < front;
else return false;
Abstract Queue peek()
Pre Condition : isempty() == false;
Post Condition : return front element;

Abstract Queue enqueue(int x)
Pre Condition: size != N
Post Condition : rear = rear + 1;
Queue[rear] = x;

Abstract Queue dequeue(int x)
Pre Condition : isempty() == false;
Post Condition : y = Queue[front];
rear = rear – 1;

```
front = front + 1;
return y;
```

**Algorithm for Queue operations using array/Linked list : (Write only the algorithm for assigned type)**

1. Start
2. Add Data entry
   - pq[capacity].age=age
   - Put pq[capacity].id=id
   - Capacity=capacity+1
3. Sort
   - Bubble sort with priority given to age
4. Delete
   - Remove the front most element
   - Shift all elements by one
   - Space=space-1
5. View
   - Print all elements along with their id and age.
6. End.

**Implementation Details:**

1) **Mention the application assigned to you and explain how you implemented the solution using the assigned type of Queue.**

**Application Assigned** – Queue for tickets priority for senior citizens and specially abled people. Perform priority queue operation for a tickets priority system. Accept booking of users and their details to insert the in queue. Display queue based on the given priority.
**Type of Queue** – Priority Queue
**Steps:**
Make a structure which holds two varaiable: id and age.

Create an array of this struct to hold the values for both the value and the age for checking priority in the queue.

1. Add Data entry- Add new entry at the end of queue (FIFO).
2. Sort-Bubble sort with priority given to age
4. Delete – Shift all elements towards the start.
5. View - Print all elements along with their id and age.

**Program source code:**

```
#include <stdio.h>
#include <stdlib.h>

struct Person
{
    int id;
    int age;
};
struct Person *priorQueue;
int space = 0;

void add(int id, int age, int size)
{
    if (space < size)
    {
        priorQueue[space].id = id;
        priorQueue[space].age = age;
        space++;
    }
    else
        printf("The Queue is full!\n");
}

void sort()
{
    for (int i = 0; i < space; i++)
    {
        for (int j = i; j < space; j++)
        {
            if (priorQueue[j].age < priorQueue[j + 1].age)
            {
```

```
            struct Person temp = priorQueue[j];
            priorQueue[j] = priorQueue[j + 1];
            priorQueue[j + 1] = temp;
         }
      }
   }
}

void delete ()
{
   if (space == 0)
      printf("The Queue is empty!\n");
   else
   {
      for (int i = 0; i < space; i++)
         priorQueue[i] = priorQueue[i + 1];
      priorQueue[space - 1].id = 0;
      priorQueue[space - 1].age = 0;
      space--;
   }
}

void view()
{
   for (int i = 0; i < space; i++)
      printf("Id: %d Age: %d\n", priorQueue[i].id , priorQueue[i].age);
}

void main()
{
   int n, t = -1;
   printf("Enter the size of the priority queue: \n");
   scanf("%d", &n);
   priorQueue = (struct Person *)malloc(sizeof(struct Person) * n);
   while (t != 4)
   {
      printf("(1) Add new entry\n");
      printf("(2) Delete Entry\n");
      printf("(3) View Queue\n");
```

```
        printf("(4) Exit\n");
        scanf("%d", &t);
        if (t == 1)
        {
            int id, age;
            printf("Enter person id: ");
            scanf("%d", &id);
            printf("Enter person age: ");
            scanf("%d", &age);
            add(id, age, n);
            sort();
        }
        else if (t == 2)
            delete ();
        else if (t == 3)
            view();
    }
}
```

**Output Screenshots:**

```
Enter the size of the priority queue:
10
(1) Add new entry
(2) Delete Entry
(3) View Queue
(4) Exit
1
Enter person id: 1
Enter person age: 20
(1) Add new entry
(2) Delete Entry
(3) View Queue
(4) Exit
1
Enter person id: 2
Enter person age: 59
(1) Add new entry
(2) Delete Entry
(3) View Queue
(4) Exit
1
Enter person id: 3
Enter person age: 25
(1) Add new entry
(2) Delete Entry
(3) View Queue
(4) Exit
1
Enter person id: 4
Enter person age: 19
(1) Add new entry
(2) Delete Entry
(3) View Queue
(4) Exit
3
Id: 2 Age: 59
Id: 3 Age: 25
Id: 1 Age: 20
Id: 4 Age: 19
(1) Add new entry
(2) Delete Entry
(3) View Queue
(4) Exit
2
(1) Add new entry
(2) Delete Entry
(3) View Queue
(4) Exit
3
Id: 3 Age: 25
Id: 1 Age: 20
Id: 4 Age: 19
(1) Add new entry
(2) Delete Entry
(3) View Queue
(4) Exit
4
```

**Applications of Queue in computer science:**

• Queue is useful in CPU scheduling, Disk Scheduling. When multiple processes require CPU at the same time, various CPU scheduling algorithms are used which are implemented using Queue data structure.

• When data is transferred asynchronously between
two processes. Queue is used for synchronization. Examples : IO Buffers, pipes, file IO, etc.

• Print Spooling. In print spooling, documents are loaded into a buffer and then the printer pulls them off the buffer at its own rate. Spooling also lets you place a number of print jobs on a queue instead of waiting for each one to finish before specifying the next one.

• Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive, First come first served.

• In real life, Call Centre phone systems will use Queues, to hold people calling them in an order, until a service representative is free

**Conclusion:- Successfully executed the given assignment. Learnt and implement priority queue.**