Experiment – 2

# IMPLEMENTATION OF QUEUE USING ARRAY

**What is Queue?**

A queue is a fundamental data structure in computer science that follows the First-In-First-Out (FIFO) principle. It is designed to hold a collection of items with specific access and manipulation rules. Queues are widely used in various computing applications for managing and processing data in a sequential order.

**Basic characteristics of Queue:**

* FIFO Principle: Items added to the queue are processed in the order they were added, resembling a line of people waiting for service.

**Operations performed on Queue:**

Two Main Operations: Queues primarily support two main operations:

* Enqueue: Adding an item to the back (end) of the queue.
* Dequeue: Removing and processing the front (first) item from the queue.

**Uses of Queue:**

* Queues are used in task scheduling systems to manage the execution order of tasks or processes. The first task enqueued is the first to be executed.
* In printer systems, incoming print jobs are enqueued in the order they are received and processed one by one.
* Queues are employed in scenarios where resources, such as CPU time or memory space, need to be allocated fairly among competing tasks.
* In distributed systems, message queues enable communication between different components or services by allowing messages to be sent and received in the order they were enqueued.

**Implementation of Queue:**

Queues can be implemented using arrays or linked lists. Linked list implementations offer dynamic resizing, while array-based implementations have a fixed size.

* The "front" of the queue is where items are dequeued, and the "back" is where items are enqueued.
* The "enqueue" operation adds an element to the back of the queue, and the "dequeue" operation removes and returns the front element.

**Time Complexity of Queue:**

* Time complexity of "enqueue" and "dequeue" operations: O(1) (constant time).
* Space complexity depends on the number of elements in the queue.

**/\*program to implement queue using array and include its main operations such as enqueue(), deque() and display queue\*/**

#include <stdio.h>

void enqueue();

void dequeue();

void display();

int queue\_array[100];

int rear = - 1;

int front = - 1;

void 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:

enqueue();

break;

case 2:

dequeue();

break;

case 3:

display();

break;

case 4:

exit(1);

default:

printf("Wrong choice \n");

}

}

}

void enqueue()

{

int add\_item;

if (rear == 100 - 1)

printf("Queue Overflow \n");

else

{

if (front == - 1)

front = 0;

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

scanf("%d", &add\_item);

rear = rear + 1;

queue\_array[rear] = add\_item;

}

}

void dequeue()

{

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

{

printf("Queue Underflow \n");

return ;

}

else

{

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

front = front + 1;

}

}

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\_array[i]);

printf("\n");

}

}