**COURSE CODE:** DJS22ITL302 **DATE:5/10/2023**

**COURSE NAME:** Data Structure Laboratory **CLASS: I1-Batch1**

**NAME:** Anish Sharma

**Experiment No. 2**

**CO/LO: CO1**

**Aim: Implements Priority Queues using array**

#### **Theory: Priority Queue** implementation using an array is one of the basic methods to implement a queue. In this element, it is inserted and deleted based on its priority. The elements in the priority queue have some priority. The priority of the element is used to determine the order in which the elements will be processed.

**Program:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#define MAX 10**

**void create\_queue();**

**void insert\_element(int);**

**void delete\_element(int);**

**void check\_priority(int);**

**void display\_priorityqueue();**

**int pqueue[MAX];**

**int front, rear;**

**void main()**

**{**

**int n, choice;**

**printf("\nEnter 1 to insert element by priority ");**

**printf("\nEnter 2 to delete element by priority ");**

**printf("\nEnter 3 to display priority queue ");**

**printf("\nEnter 4 to exit");**

**create\_queue();**

**while (1)**

**{**

**printf("\nEnter your choice : ");**

**scanf("%d", &choice);**

**switch(choice)**

**{**

**case 1:**

**printf("\nEnter element to insert : ");**

**scanf("%d",&n);**

**insert\_element(n);**

**break;**

**case 2:**

**printf("\nEnter element to delete : ");**

**scanf("%d",&n);**

**delete\_element(n);**

**break;**

**case 3:**

**display\_priorityqueue();**

**break;**

**case 4:**

**exit(0);**

**default:**

**printf("\n Please enter valid choice");**

**}**

**}**

**}**

**void create\_queue()**

**{**

**front = rear = -1;**

**}**

**void insert\_element(int data)**

**{**

**if (rear >= MAX - 1)**

**{**

**printf("\nQUEUE OVERFLOW");**

**return;**

**}**

**if ((front == -1) && (rear == -1))**

**{**

**front++;**

**rear++;**

**pqueue[rear] = data;**

**return;**

**}**

**else**

**check\_priority(data);**

**rear++;**

**}**

**void check\_priority(int data)**

**{**

**int i,j;**

**for (i = 0; i <= rear; i++)**

**{**

**if (data >= pqueue[i])**

**{**

**for (j = rear + 1; j > i; j--)**

**{**

**pqueue[j] = pqueue[j - 1];**

**}**

**pqueue[i] = data;**

**return;**

**}**

**}**

**pqueue[i] = data;**

**}**

**void delete\_element(int data)**

**{**

**int i;**

**if ((front==-1) && (rear==-1))**

**{**

**printf("\nEmpty Queue");**

**return;**

**}**

**for (i = 0; i <= rear; i++)**

**{**

**if (data == pqueue[i])**

**{**

**for (; i < rear; i++)**

**{**

**pqueue[i] = pqueue[i + 1];**

**}**

**pqueue[i] = -99;**

**rear--;**

**if (rear == -1)**

**front = -1;**

**return;**

**}**

**}**

**printf("\n%d element not found in queue", data);**

**}**

**void display\_priorityqueue()**

**{**

**if ((front == -1) && (rear == -1))**

**{**

**printf("\nEmpty Queue ");**

**return;**

**}**

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

**{**

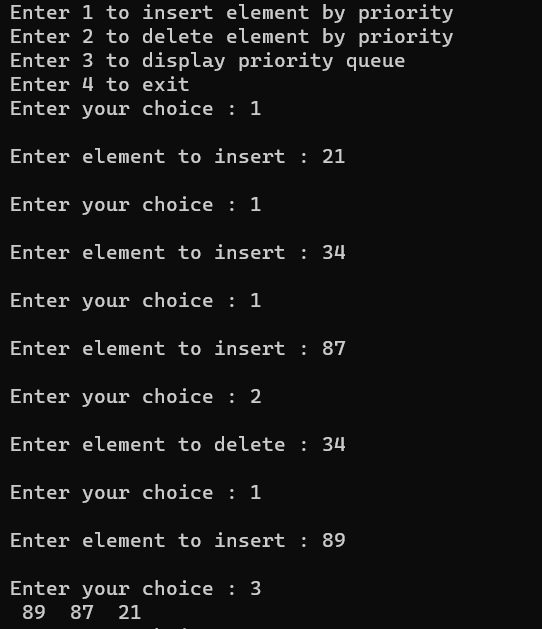
**printf(" %d ", pqueue[front]);**

**}**

**front = 0;**

**}**

**Output screenshots:**

****

#### **Conclusion:**

a priority queue is a versatile data structure that extends the functionality of a standard queue by assigning a priority to each element. Elements with higher priority are dequeued and processed before elements with lower priority.

**REFERENCES:**

Tutorialpoints