Experiment No. : 4

Title: Implementation of Static Priority Queue

Batch: A3 **Roll No.: 16010423099** **Experiment No.: 4**

Aim: Write a menu driven program to implement a static priority queue supporting following operations.

1. Create empty queue,
2. Insert an element on the queue,
3. Delete an element from the queue,
4. Display front, rear element or
5. Display all elements of the queue.

Resources Used: Turbo C/ C++ editor and compiler.

Theory:

Priority Queue -

A **priority queue** is a specialized data structure that operates similarly to a regular queue but with an important twist: each element has a priority associated with it. In a priority queue, elements are dequeued based on their priority rather than their order in the queue.

Algorithm :

Algorithm for Insertion and Deletion in a Priority Queue:

Insertion (Insert):
Input: Element and its priority.
Create a new node with the given element and priority.
Find the correct position in the queue based on the priority.
Traverse the queue to find the appropriate spot.
Insert the node at that position.
Adjust pointers as necessary.

Deletion (Remove):
Check if the queue is empty. If so, return an error.
Identify the highest priority element (usually at the front).
Remove that element from the queue.
Adjust the front pointer to the next element in line.
Return the removed element.

Program:

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

#define MAX 10

typedef struct {
    int data[MAX];
    int priority[MAX];
    int size;
} PriorityQueue;

void createQueue(PriorityQueue* pq) {
    pq->size = 0;
}

int isEmpty(PriorityQueue* pq) {
    return pq->size == 0;
}

int isFull(PriorityQueue* pq) {
    return pq->size == MAX;
}

void insert(PriorityQueue* pq, int element, int priority) {
    if (isFull(pq)) {
        printf("Queue's packed! Can't squeeze in %d.\n", element);
        return;
    }

    int i = pq->size - 1;
    while (i >= 0 && priority > pq->priority[i]) {
        pq->data[i + 1] = pq->data[i];
        pq->priority[i + 1] = pq->priority[i];
        i--;
    }
    pq->data[i + 1] = element;
    pq->priority[i + 1] = priority;
    pq->size++;
    printf("Inserted %d with priority %d—welcome aboard!\n", element, priority);
}

int delete(PriorityQueue* pq) {
    if (isEmpty(pq)) {
        printf("Nothing to delete...\n");
        return -1;
    }
```

```
    return pq->data[--pq->size];
}

void displayFront(PriorityQueue* pq) {
    if (isEmpty(pq)) {
        printf("Queue's empty!\n");
    } else {
        printf("Front element: %d\n", pq->data[0]);
    }
}

void displayRear(PriorityQueue* pq) {
    if (isEmpty(pq)) {
        printf("Queue's empty; no rear to show!\n");
    } else {
        printf("Rear element: %d\n", pq->data[pq->size - 1]);
    }
}

void displayAll(PriorityQueue* pq) {
    if (isEmpty(pq)) {
        printf("Queue's empty...\n");
        return;
    }
    printf("Queue elements (priority):\n");
    for (int i = 0; i < pq->size; i++) {
        printf("Element: %d, Priority: %d\n", pq->data[i], pq->priority[i]);
    }
}

int main() {
    PriorityQueue pq;
    createQueue(&pq);
    int choice, element, priority;

    do {
        printf("\n--- Priority Queue Menu ---\n");
        printf("1. Create empty queue\n");
        printf("2. Insert an element\n");
        printf("3. Delete an element\n");
        printf("4. Display front element\n");
        printf("5. Display rear element\n");
        printf("6. Display all elements\n");
        printf("7. Exit\n");
        printf("Your move: ");
        scanf("%d", &choice);

        switch (choice) {
```

```
            case 1:
                createQueue(&pq);
                printf("A new queue starts.\n");
                break;
            case 2:
                printf("Enter element: ");
                scanf("%d", &element);
                printf("Enter priority: ");
                scanf("%d", &priority);
                insert(&pq, element, priority);
                break;
            case 3:
                element = delete(&pq);
                if (element != -1) {
                    printf("Farewell to %d—out it goes!\n", element);
                }
                break;
            case 4:
                displayFront(&pq);
                break;
            case 5:
                displayRear(&pq);
                break;
            case 6:
                displayAll(&pq);
                break;
            case 7:
                printf("Exiting—until next time!\n");
                break;
            default:
                printf("Don't try to be extra smart. Pick a valid option.\n");
        }
    } while (choice != 7);

    return 0;
}
```

Results:

```
D:\MinGW\stuff\16010423099_EXP4_DS.exe

--- Priority Queue Menu ---
1. Create empty queue
2. Insert an element
3. Delete an element
4. Display front element
5. Display rear element
6. Display all elements
7. Exit
Your move: 1
A new queue starts.

--- Priority Queue Menu ---
1. Create empty queue
2. Insert an element
3. Delete an element
4. Display front element
5. Display rear element
6. Display all elements
7. Exit
Your move: 2
Enter element: 5
Enter priority: 1
Inserted 5 with priority 1ùwelcome aboard!

--- Priority Queue Menu ---
1. Create empty queue
2. Insert an element
3. Delete an element
4. Display front element
5. Display rear element
6. Display all elements
7. Exit
Your move: 6
Queue elements (priority):
Element: 5, Priority: 1

--- Priority Queue Menu ---
1. Create empty queue
2. Insert an element
3. Delete an element
4. Display front element
5. Display rear element
6. Display all elements
7. Exit
Your move: 7
Exitingùuntil next time!

Process returned 0 (0x0)   execution time : 49.029 s
```

Outcome:

CO2: Apply linear and non-linear data structure in application development.

Conclusion:

Program executed successfully implementing a priority queue using linear data structures and switch case.

Grade: AA / AB / BB / BC / CC / CD /DD:

Signature of faculty in-charge with date :

References:

Books/ Journals/ Websites:

- Y. Langsam, M. Augenstin and A. Tannenbaum, “Data Structures using C”, Pearson Education Asia, 1st Edition, 2002.