DSA - Experiment 6

Name: Divyesh Khunt **Sapid:**60009210116 **Batch: A/3**

Aim: To create a queue in c programming and learn its abstract data Types.

Theory:

Queue is a linear data structure which follows a order (FIFO First In First Out) or LILO (Last In Last Out) in which operations are performed.

There are 4 operations in queues insert, delete, peek, display

1. INSERT
   When we insert an element in a queue then the operation is known as a push. The next element added is added above the previous one.
2. DELETE
   When we delete an element from the queue, the operation is known as a pop. Here the topmost element is deleted.
3. PEEK
   It returns the element at the front position.
4. DISPLAY
   It prints all the elements available in the queue.

OVERFLOW: If the queue is full then the overflow condition occurs.

UNDERFLOW: If the queue is empty means that no element exists in the queue, this state is known as an underflow state.

Time Complexity of Queue Operations

As mentioned above, only a single element can be accessed at a time in Queues.

While performing push () and pop() operations on the queue, it takes **O(1)** time.

CODE:

```c
#include <stdio.h>
#include<conio.h>
# define SIZE 100
void insert();
void delet();
void show();
void peek();
int a[SIZE];
int Rear = - 1;
int Front = - 1;
main()
{
    int ch;
    while (1)
    {
        printf("\n1.insert Operation\n");
        printf("2.delete Operation\n");
        printf("3.Display the Queue\n4.peek\n");
        printf("5.Exit\n");
        printf("Enter your choice of operations : ");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
            insert();
            break;
            case 2:
            delet();
            break;
            case 3:
            show();
            break;
            case 4:
            peek();
            case 5:
            break;
            default:
            printf("Incorrect choice \n");
        }
    }
}
```

```c
void insert()
{
    int insert_item;
    if (Rear == SIZE - 1)
       printf("Overflow \n");
    else
    {
        if (Front == - 1)

        Front = 0;
        printf("\nElement to be inserted in the Queue\n : ");
        scanf("%d", &insert_item);
        Rear = Rear + 1;
        a[Rear] = insert_item;
    }
}

void delet()
{
    if (Front == - 1 || Front > Rear)
    {
        printf("Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from the Queue: %d\n", a[Front]);
        Front = Front + 1;
    }
}
```

```c
void show()
{

    if (Front == - 1)
        printf("Empty Queue \n");
    else
    {
        printf("Queue: \n");
        for (int i = Front; i <= Rear; i++)
            printf("%d ", a[i]);
        printf("\n");
    }
}

void peek()
{
if (Front == - 1 || Front > Rear)
    {
        printf("Underflow \n");
        return ;
}
else
    {
      printf("The element at the front is %d",a[Front]);
    }
}
```

OUTPUTS:

INSERT

```
1.insert Operation
2.delete Operation
3.Display the Queue
4.peek
5.Exit
Enter your choice of operations : 1

Element to be inserted in the Queue
 : 87

1.insert Operation
2.delete Operation
3.Display the Queue
4.peek
5.Exit
Enter your choice of operations : 1

Element to be inserted in the Queue
 : 36

1.insert Operation
2.delete Operation
3.Display the Queue
4.peek
5.Exit
Enter your choice of operations : 1

Element to be inserted in the Queue
 : 23
```

DISPLAY

```
1.insert Operation
2.delete Operation
3.Display the Queue
4.peek
5.Exit
Enter your choice of operations : 3
Queue:
87 36 23
```

DELETE

```
1.insert Operation
2.delete Operation
3.Display the Queue
4.peek
5.Exit
Enter your choice of operations : 2
Element deleted from the Queue: 87
```

PEEK

```
1.insert Operation
2.delete Operation
3.Display the Queue
4.peek
5.Exit
Enter your choice of operations : 4
The element at the front is 36
```

APPLICATIONS OF QUEUE:

- CPU Scheduling
- Disk Scheduling
- Asynchronous data transfer between processors such as File IO, etc.
- Breadth-First Search Algorithm (BFS)

CONCLUSION:

Thus, in this article, we have understood the concept of Queue data structure and its implementation using Arrays in C.