TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

PURWANCHAL CAMPUS

Dharan-8

A Lab Report On: Queue

Submitted By

Name: Dhiraj KC
Roll No. : PUR077BEI014
Faculty: Electronics, Communication and Information
Group: A

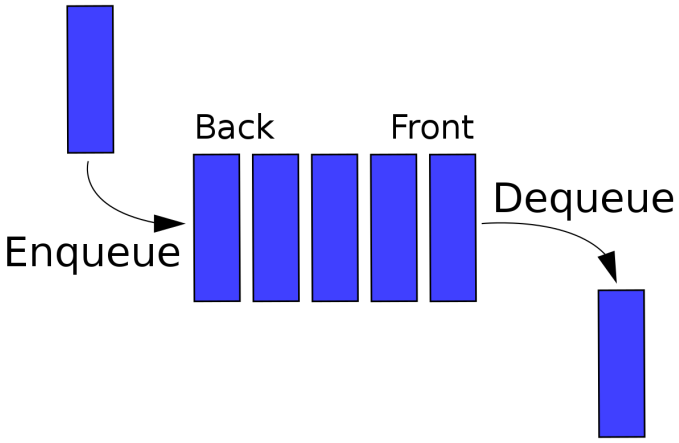
Submitted To

Department Of Electronics and Computer Engineering
Checked By: Pukar Karki

TITLE: ALGORITHM OF IMPLEMENTATION OF SIMPLE QUEUE

THEORY

A Queue is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order. We define a queue to be a list in which all additions to the list are made at one end, and all deletions from the list are made at the other end.



The two major operations of Queue are:

1. Enqueue: It is used to push the elements in the queue data structure.
2. Dequeue: It is used to pop out the elements from the front.

Types of Queue:

1. Simple Queue:
   In Simple Queue, Insertion and Deletion can be only done from one end of the Queue

2. Circular Queue
   In Circular Queue, the Last Element points to the First Element making a Circular Link.

3. Priority Queue
   In Priority Queue, each element is associated with a priority served automatically.

Eg: Ascending Priority Queue, Descending Priority Queue etc.

4. Double Ended Queue
   In Double Ended Queue, Insertion and Deletion can be performed from both ends (rear and front) of the Queue.

ALGORITHM

1. Start
2. Initialize variables (front, back)
3. Initialize an array with size n
4. Create a function enqueue() to insert element in queue.
5. Create a function dequeue() to delete element.
6. Create a function to print the element in queue.
7. Get User input from main() function and proceed the use cases of function mentioned above.
8. Stop

PROGRAM

```
#include <stdio.h>
#define SIZE 100
int inp_arr[SIZE];
int Rear = -1;
int Front = -1;

void enqueue()
{
 int insert_item;
 if (Rear == SIZE - 1)
  printf("Overflow \n");
 else
 {
  if (Front == -1)
   Front = 0;
  printf("Element to be inserted in the Queue\n : ");
  scanf("%d", &insert_item);
  Rear = Rear + 1;
  inp_arr[Rear] = insert_item;
 }
}

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

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

```
main()
{
 int ch;
 while (1)
 {
  printf("1.Enqueue Operation\n");
  printf("2.Dequeue Operation\n");
  printf("3.Display the Queue\n");
  printf("4.Exit\n");
  printf("Enter your choice of operations : ");
  scanf("%d", &ch);
  switch (ch)
  {
  case 1:
   enqueue();
   break;
  case 2:
   dequeue();
   break;
  case 3:
   show();
   break;
  case 4:
   exit(0);
  default:
   printf("Incorrect choice \n");
  }
 }
}
```

OUTPUT

```
1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 1
Element to be inserted in the Queue: 10

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 1
Element to be inserted in the Queue: 20

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 3
Queue:
10 20

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations : 2
Element deleted from the Queue: 10

1.Enqueue Operation
2.Dequeue Operation
3.Display the Queue
4.Exit
Enter your choice of operations: 3
Queue:
20
```