Apex College

BCIS Program

Affiliated to Pokhara University

Data Structure & Algorithms
Lab Report

2. Queue Operations: Enqueue & Dequeue

Date: 08-05-2022

Submitted by:
Ishwor Shrestha
Roll no.: 2018-BCIS-414

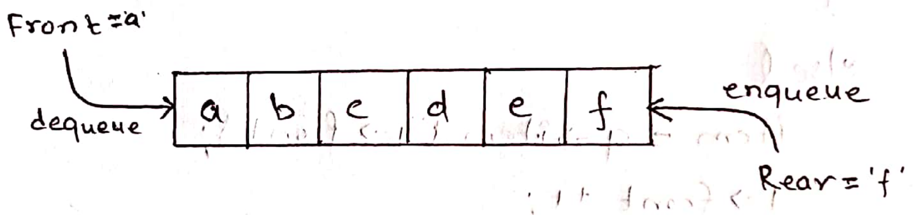
Submitted to:
Pravakar Ghimire, &
Anmol Shrestha
Apex College

#Lab1 Objectives

- To understand queue and its operations
- To create program based on algorithms of enqueue and dequeue.

#Introduction

Queue is an ordered collection of items which insert (enqueue) data from rear side (i.e. tail) and remove (dequeue) data from front side (ie. head) also called the FIFO principle. So that the data item stored first will be accessed first



#A menu driven program to demonstrate the operations in the stack.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10

struct queue {
    int items [MAX]
    int rear, front;
};

typedef struct queue queue;
```

```
void enqueue(queue *q, int item){
    if (q->rear == MAX-1)
        printf("Queue is full.\n");
    else {
        q-> rear ++;
        q-> items [q-> rear] = item;
    }
}

int dequeue (queue *q){
    int item = -1;
    if(q-> front > q-> rear){
        printf("Queue is empty.\n");
    }
    else {
        item = q->item [q-> front];
        q-> front ++;
    }
    return item;
}

void display (queue *q) {
    int i;
    printf("Elements in Queue:\n");
    for(i=q->rear; i>=q->front; i--)
        printf("%d\n", q->items[i]);
}

void main () {
    queue q;
    int ch, item;
    q.front=0;
    q. rear =-1;
```

~~void main()~~

```
while (1) {
    printf("Enter your choice :\n");
    printf("1. Enqueue \n 2. Dequeue \n 3. Display \n 4.Exit\n");
    Scanf("%d", &ch);

    switch (ch) {

      case 1 :
        printf("Enter data : ");
        scanf("%d", &item);
        enqueue (&q, item);
        break;

      case 2 :
        item = dequeue (&q);
        printf("%d" is dequeue.\n", item);
        break;

      case 3:
        display (&q);
        break;

      case 4:
        exit (0);

      default:
        printf("Invalid choice.\n");

    }
  }
}
```

Activities

- Created 4 different menu items similar to stack.

① Dequeue
  - used to remove data from the front of the queue following FIFO sequence.
  - check whether the queue is ~~full~~ empty or not
    i.e. if (Rear < ~~[illegible]~~ Front)
      printf("Queue is ~~full~~ empty" & exit.

• otherwise, queue is not ~~full~~ empty
  - ~~increment the value~~
  - set, item = queue [front];
  - increment; value of front
  - return item

③ Enqueue
- used to insert data from the rear side of the queue following FIFO sequence.
  • check whether queue is full or not
    i.e. if (rear == MAX-1)
        printf ("Queue is full" & exit
  • Otherwise, queue is not full
    - increment : value of Rear
        i.e. rear++;
    - enqueue new element into rear position of queue
        i.e. queue [rear] = item

④ Display

⑤ Exit

#Conclusion

- Learned operations of queue i.e. enqueue and dequeue.
- enqueue operation used to insert item (data) from rear side of queue & dequeue operation used to remove data from front side of queue.