Apex College

BCIS Program

Affiliated to Pokhara University

Data Structure & Algorithms
Lab Report

7: Dynamic Implementation of Queue using Linked List

Date: 06-06-2022

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

Submitted to:
Pravakar Ghimire, &
Anmol Shrestha
Apex College

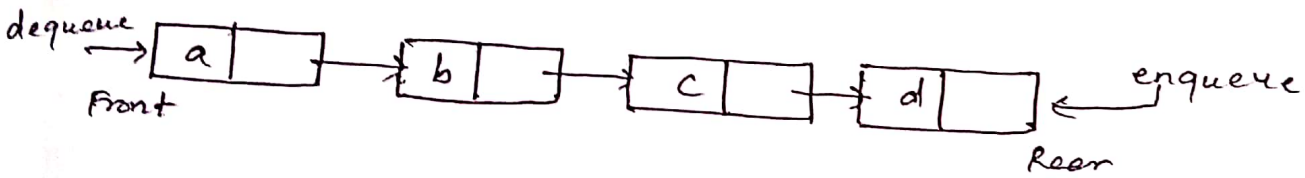
Lab 7 Objectives

- To implement queue using linked list
- To allocate memory dynamically as per requirement
- To implement FIFO sequence.

Introduction

Queue is an ordered collection of items which follows FIFO principle to enqueue data from rear side & dequeue data (node) from front side of queue.

Queue can be implemented using linked list, where it allows to grow the queue as per requirement i.e. memory can be allocated dynamically. This performs on Heap Segment.



A program to demonstrate queue using linked list

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

struct node {
   int data;
   struct node *next;
};
typedef struct node node;
struct node *front= NULL;
struct node * rear = NULL;
```

```
void enqueue (int value){
    struct node *temp
    temp = (node *) malloc (sizeof (node));
    temp -> data = item;
    temp -> next = NULL;
    if ((front == NULL) && (rear = NULL))
        front = rear = temp;
    else {
        rear -> next = temp;
        rear = temp;
    }
    printf("Node is enqueued. \n\n");
}

int dequeue () {
    if (front == NULL) {
        printf("\n Underflow. queue is empty *\n");
        return -1;
    }
    else {
        node *temp = front;
        int item = front -> data;
        front = front -> next;
        free (temp);
        return item;
    }
}

void display () {
    node *temp;
    if ((front == NULL) && (rear == NULL))
        printf("\n Queue is empty. \n");
```

```
    else {
        printf("The queue is \n");
        temp = front;
        while (temp) {
            printf("%d \t", temp -> data);
            temp = temp -> next;
        }
        printf("NULL \n");
    }
}

int main() {
    int ch, item;
    while (ch)
        printf("Enter your choice: \n");
        printf("1. Enqueue \t 2. Dequeue \t 3. Display \t 4. Exit \n");
        scanf("%d", &ch);
    switch (ch) {
        case 1:
            printf("\n Enter data to enqueue with node: \n");
            scanf("%d", &item);
            enqueue(item);
            break;
        case 2:
            printf("Dequeued data is: %d \n", dequeue());
            break;
        case 3:
            display();
            break;
        case 4:
            exit(0);
            break;
```

```
        default:
            printf("\n Invalid choice .\n");
        }
    }
    return 0;
}
```

Activities

① Enqueue operation using Insertion operation from end of singly linked list

② Dequeue operation using Deletion from beginning of singly linked list.

③ Display

④ Exit

Conclusion

A queue can be represent dynamically ~~memory~~ allocating memory ~~with~~ by using different algorithm of linked list.