INDEX

Ex. No.:4

Date :

Implementation of Linear Queue Using Arrays

Aim

To write a C-program to implement linear queue data structure using arrays.

Operations on queue:

MakeEmpty(q): To make q as an empty queue
Enqueue(q, x): To insert an item x at the rear of the queue, this is also called by names add, insert.
Dequeue(q): To delete an item from the front of the queue q. This is also known as Delete, Remove.
IsFull(q): To check whether the queue q is full.
IsEmpty(q): To check whether the queue q is empty
Traverse (q): To read entire queue that is display the content of the queue.

Algorithm for insertion an item in queue:

```
1. Initialize front=0 and rear=-1
        if rear>=MAXSIZE-1
            print "queue overflow" and return
        else
            set rear=rear+1
            queue[rear]=item
2. End.
```

Algorithm to delete an element from the queue:

```
1. if rear<front
        print "queue is empty" and return
   else
       item=queue[front++]
2. end
```

Declaration of a Queue:

```
# define MAXQUEUE 100 /* size of the queue items*/
struct queue
{
    int front;
    int rear;
    int items[MAXQUEUE];
};
typedef struct queue qt;
```

Defining the operations of linear queue:

1. ***The MakeEmpty function:***

```
void makeEmpty(qt  *q)
{
    q->rear=-1;
   q->front=0;
}
```

2. ***The IsEmpty function:***

```
int IsEmpty(qt *q)
{
   if(q->rear<q->front)
       return 1;
  else
      return 0;
}
```

3. ***The Isfull function:***

```
int IsFull(qt *q)
{
       if(q->rear==MAXQUEUEZIZE-1)
         return 1;
      else
         return 0;
}
```

Program

```
/*Array implementation of linear queue*/
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define MAXSIZE 100

struct queue
{
    int items[MAXSIZE];
    int rear;
    int front;
};

typedef struct queue qt;
void insert(qt*);
void delet(qt*);
void display(qt*);
void make_empty(qt*);



int main()
{
```

```
    int ch;
    qt *q;
    make_empty(q);
    //clrscr();
    do
    {
    printf("\nMenu for program:\n");
    printf("1:insert\n2:delete\n3:display\n4:exit\n");

        printf("Enter your choice\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                insert(q);
                break;
            case 2:
                delet(q);
                break;
            case 3:
                display(q);
                break;
            case 4:
                exit(1);
                break;
            default:
            printf("Your choice is wrong\n");
        }
    }while(ch<5);
    getch();
    return 0;
}
/**********Make empty queue************/
void make_empty(qt *q)
{
    q->rear=-1;
    q->front=0;
}
/**********insert function************/
 void insert(qt *q)
{
    int d;
    printf("Enter data to be inserted\n");
    scanf("%d",&d);
    if(q->rear==MAXSIZE-1)
    {
        printf("Queue is full\n");
    }
    else
    {
        q->rear++;
        q->items[q->rear]=d;
    }
}

/***********delete function****************/
```

```
void delet(qt *q)
{
    int d;
    if(q->rear<q->front)
    {
        printf("Queue is empty\n");
    }
    else
    {
        d=q->items[q->front];
        q->front++;
        printf("Deleted item is:");
        printf("%d\n",d);
    }
}
/**************display function***********/
void display(qt *q)
{
    int i;
    if(q->rear<q->front)
    {
        printf("Queue is empty\n");
    }
    else
    {
        for(i=q->front;i<=q->rear;i++)
        {
            printf("%d\t",q->items[i]);
        }
    }
}
```