Data Structures using C

25th March 2022

Parth Nikam
20070123120
E&TC – B3

Aim: - Studying and coding on queue.

Objective: - To perform enqueue, dequeue, and display operations on queue in C language

Code: -

E&TC-B3
20070123120
Parth Nikam

Queue

```
#include <stdio.h>
# define SIZE 100

void enqueue(void);
void dequeue(void);
void show(void);
int int_arr[SIZE];
int Rear = -1;
int Front = -1;

int main(){
    int ch;
    while(1){
        printf("\n1.Enqueue \n2. Dequeue \n3. Display Queue
                \n4. Exit \n Enter your choice (1-4):");
        scanf("%d", &ch);

        switch(ch){
          case 1:
              Equeue();
              break;
          Case 2:
              dequeue();
              break;
          case 3:
              Show();
              break;
          case 4: exit(0);
          default:
              printf("Incorrect choice);
          }
    }
}
```

Data Structures using C

25th March 2022

Parth Nikam
20070123120
E&TC – B3

E&TC-B3
20070123120
Parth Nikam

Queue

```
void enqueue () {
    int insert_item;
    if (Rear = SIZE -1)
        printf (" Overflow \n");
    else {
        if (front = = -1) {
            front =0;
            printf(" Element ~~to~~ be inserted: ");
            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 to be deleted %d \n", inp_arr[front]);
        front = front +1;
    }
}

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

Data Structures using C

25th March 2022

Parth Nikam
20070123120
E&TC – B3

Screenshot: -

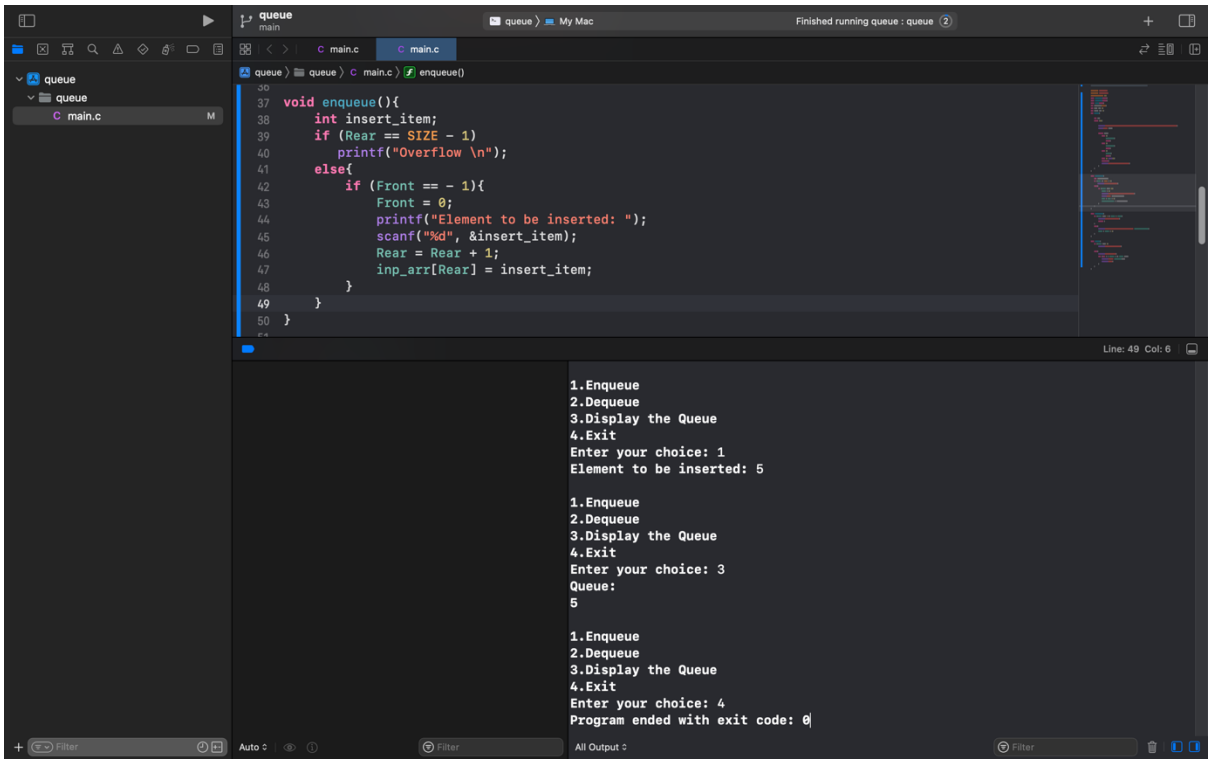
```c
// Parth Nikam PRN: - 20070123120 E&TC - B3 25th March 2022

#include <stdio.h>
#include <stdlib.h>
# define SIZE 100
void enqueue(void);
void dequeue(void);
void show(void);
int inp_arr[SIZE];
int Rear = - 1;
int Front = - 1;
int main(){

    int ch;
    while (1){

        printf("\n1.Enqueue\n2.Dequeue\n3.Display the Queue\n4.Exit\nEnter your choice: ");
        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");
        }
    }
}

void enqueue(){

    int insert_item;
```

```c
void enqueue(){
    int insert_item;
    if (Rear == SIZE - 1)
       printf("Overflow \n");
    else{
        if (Front == - 1){
            Front = 0;
            printf("Element to be inserted: ");
            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 to be deleted: %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");
        }
    }
}
```

Data Structures using C

25th March 2022

Parth Nikam
20070123120
E&TC – B3



Result: - A queue in C is basically a linear data structure to store and manipulate the data elements. It follows the order of First In First Out (FIFO). In queues, the first element entered into the array is the first element to be removed from the array.