Experiment No. 2

Aim: Implementation of queue using Array for real-world application

Objectives:

1) To introduce the concepts of data structures and analysis procedure
2) To conceptualize linear data structures and its implementation for various real-world applications.

Theory:

* Introduction to linear and non-linear data structure:

Linear Data Structure - In linear data structure, data elements are arranged sequentially or linearly where the elements are arranged to its previous and next adjacent element.

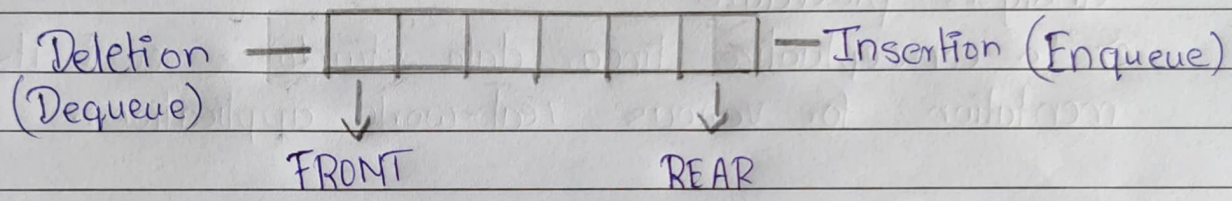
— Examples of linear data structures are array, stack, queue, lists, etc.

Non-linear Data Structure - In non-linear data structure, data elements are not arranged sequentially or linearly. Here, we cannot traverse all the elements in a single run.

Examples of non-linear data structures are graphs and trees.

* Introduction to Queue :-

A queue is a linear data structure which follows a particular order in which the operations are performed. The order is FIFO.
In a queue, new elements are added ~~to queue~~ from one end called REAR and elements are always removed from other ~~hand~~ end called FRONT end.



* Operations in Queue -

1) Enqueue - Adds an item in a queue
2) Dequeue - Removes an item from a queue
3) Display - Returns all the items from a queue.

Algorithm -

① INSERT (Q, F, R, N, Y) : Given 'F' & 'R', pointers to front and rear elements of queue 'Q' having 'N' elements, element 'Y' insertion in queue 'Q'.

1. If $R \geq N$
   then write ('Overflow'), Return
2. [Increment rear pointer] $R \leftarrow R+1$
3. [Insert element] $Q[R] \leftarrow Y$
4. [Is front pointer properly set?]
   If $F=0$, then $F \leftarrow 1$, Return

(2) DELETE (Q, F, R) : Given 'F' and 'R' pointers to front and rear elements of queue 'Q', element 'y' is to be deleted.

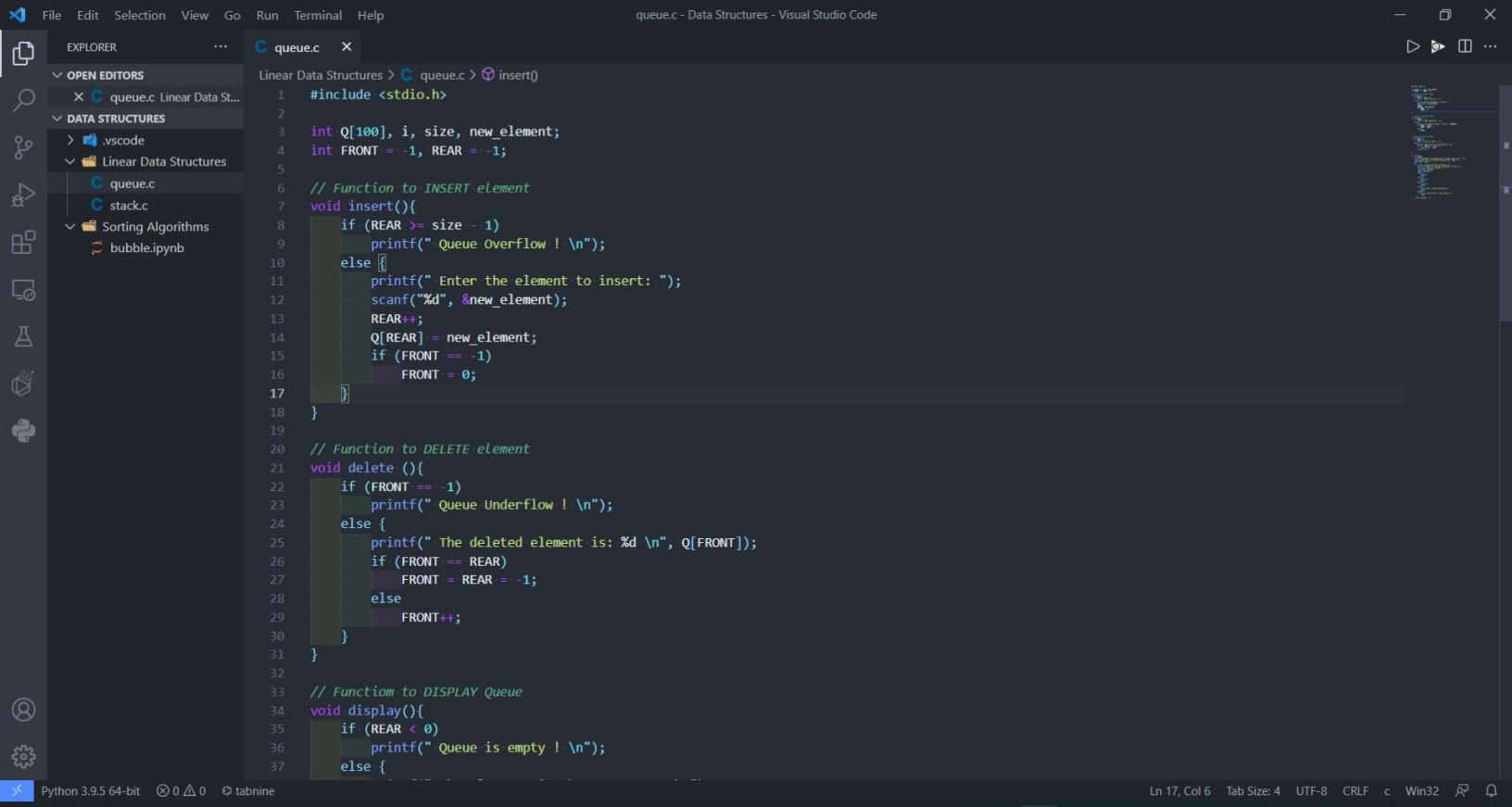
1. If F = 0
   then write ('Underflow')
   Return (0)
2. [Delete element] $y \leftarrow Q[F]$
3. [Queue empty]
   if F = R
   then $F \leftarrow R \leftarrow 0$
   else $F \leftarrow F + 1$ (increment front pointer)
4. [Return element] Return [Y]

Example -

1) Luggage checking machine at railway stations and airports.
2) ~~The person who calls~~
2) Phone answering system.
3) People standing at a ticket counter.

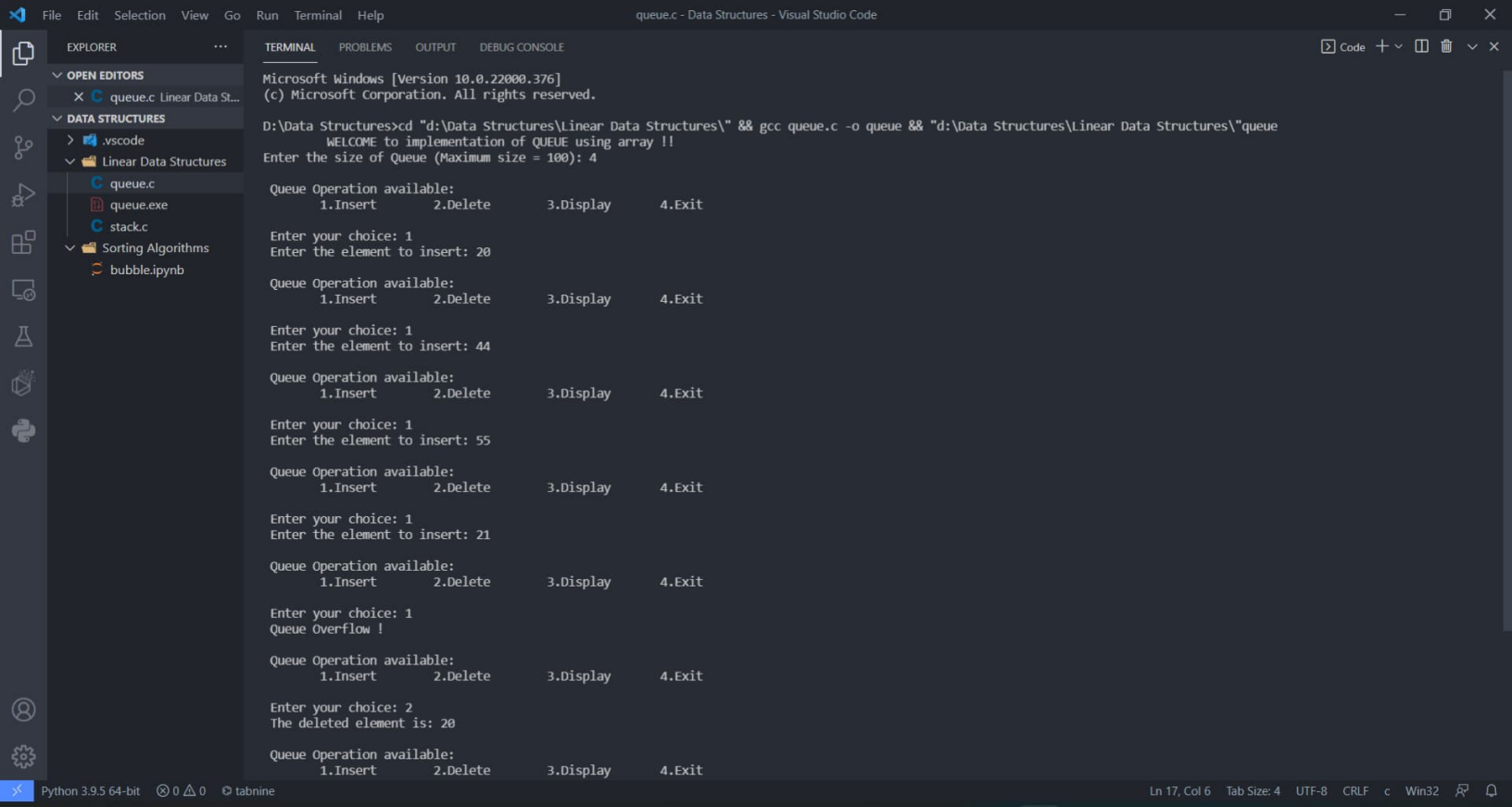
Conclusion: In this experiment, we learnt about queues in data structures. We implemented ~~stack~~ queue and performed its basic operations like enqueue and dequeue. We came across the examples where stack is implemented in real-world.

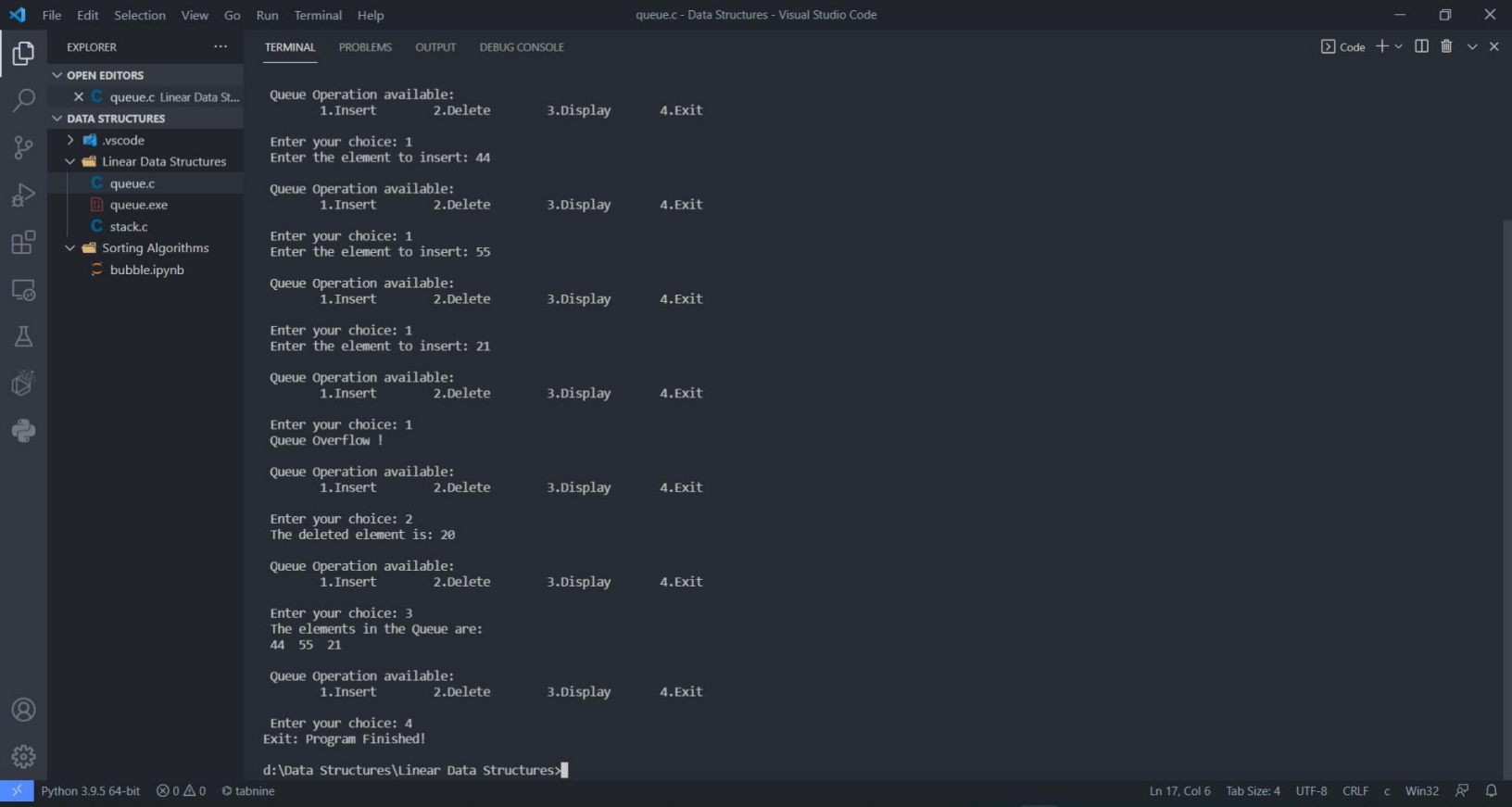
Output: Apply the concepts of stack for real-world application.


File Edit Selection View Go Run Terminal Help
queue.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
queue.c Linear Data St...
DATA STRUCTURES
.vscode
Linear Data Structures
queue.c
stack.c
Sorting Algorithms
bubble.ipynb
queue.c
Linear Data Structures > queue.c > insert()
#include <stdio.h>
int Q[100], i, size, new_element;
int FRONT = -1, REAR = -1;
// Function to INSERT element
void insert(){
if (REAR >= size - 1)
printf(" Queue Overflow ! \n");
else {
printf(" Enter the element to insert: ");
scanf("%d", &new_element);
REAR++;
Q[REAR] = new_element;
if (FRONT == -1)
FRONT = 0;
}
}
// Function to DELETE element
void delete (){
if (FRONT == -1)
printf(" Queue Underflow ! \n");
else {
printf(" The deleted element is: %d \n", Q[FRONT]);
if (FRONT == REAR)
FRONT = REAR = -1;
else
FRONT++;
}
}
// Functiom to DISPLAY Queue
void display(){
if (REAR < 0)
printf(" Queue is empty ! \n");
else {
Python 3.9.5 64-bit
0 0
tabnine
Ln 17, Col 6
Tab Size: 4
UTF-8
CRLF
c
Win32

```c
    else {
        printf(" The elements in the Queue are: \n");
        for (i = FRONT; i < size; i++)
            printf(" %d ", Q[i]);
        printf("\n");
    }
}

void main() {
    int choice;
    printf("\t WELCOME to implementation of QUEUE using array !! \n");
    printf("Enter the size of Queue (Maximum size = 100): ");
    scanf("%d", &size);
    do{
        printf("\n Queue Operation available: \n");
        printf("\t1.Insert \t2.Delete \t3.Display \t4.Exit \n");
        printf("\n Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
        case 1:
            insert();
            break;
        case 2:
            delete ();
            break;
        case 3:
            display();
            break;
        case 4:
            printf("Exit: Program Finished!\n");
            break;
        default:
            printf("Please enter a valid choice!\n");
            break;
        }
    } while (choice != 4);
}
```


File Edit Selection View Go Run Terminal Help
queue.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
queue.c Linear Data St...
DATA STRUCTURES
.vscode
Linear Data Structures
queue.c
queue.exe
stack.c
Sorting Algorithms
bubble.ipynb
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
Code
Microsoft Windows [Version 10.0.22000.376]
(c) Microsoft Corporation. All rights reserved.
D:\Data Structures>cd "d:\Data Structures\Linear Data Structures\" && gcc queue.c -o queue && "d:\Data Structures\Linear Data Structures\"queue
WELCOME to implementation of QUEUE using array !!
Enter the size of Queue (Maximum size = 100): 4
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter the element to insert: 20
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter the element to insert: 44
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter the element to insert: 55
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter the element to insert: 21
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Queue Overflow !
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 2
The deleted element is: 20
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Python 3.9.5 64-bit
tabnine
Ln 17, Col 6
Tab Size: 4
UTF-8
CRLF
c
Win32


File Edit Selection View Go Run Terminal Help
queue.c - Data Structures - Visual Studio Code
EXPLORER
OPEN EDITORS
queue.c Linear Data St...
DATA STRUCTURES
.vscode
Linear Data Structures
queue.c
queue.exe
stack.c
Sorting Algorithms
bubble.ipynb
TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE
Code
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter the element to insert: 44
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter the element to insert: 55
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter the element to insert: 21
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Queue Overflow !
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 2
The deleted element is: 20
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 3
The elements in the Queue are:
44 55 21
Queue Operation available:
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 4
Exit: Program Finished!
d:\Data Structures\Linear Data Structures>
Python 3.9.5 64-bit
tabnine
Ln 17, Col 6
Tab Size: 4
UTF-8
CRLF
c
Win32