UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 9

Queues

Submitted by:
Gabijan, Rhovic M.

Instructor:
Engr. Maria Rizette H. Sayo

October, 11, 2025

I. Objectives

Introduction

Another fundamental data structure is the queue. It is a close "the same" of the stack, as a queue is a collection of objects that are inserted and removed according to the first-in, first-out (FIFO) principle. That is, elements can be inserted at any time, but only the element that has been in the queue the longest can be next removed.

The Queue Abstract Data Type

Formally, the queue abstract data type defines a collection that keeps objects in a sequence, where element access and deletion are restricted to the first element in the queue, and element insertion is restricted to the back of the sequence. This restriction enforces the rule that items are inserted and deleted in a queue according to the first-in, first-out (FIFO) principle. The queue abstract data type (ADT) supports the following two fundamental methods for a queue Q:

Q.enqueue(e): Add element e to the back of queue Q.

Q.dequeue( ): Remove and return the first element from queue Q;
an error occurs if the queue is empty.

The queue ADT also includes the following supporting methods (with first being analogous to the stack's top method):

Q.first(): Return a reference to the element at the front of queue Q, without removing it;
an error occurs if the queue is empty.

Q.is empty( ): Return True if queue Q does not contain any elements.

len(Q): Return the number of elements in queue Q; in Python, we implement this with the special method len .

This laboratory activity aims to implement the principles and techniques in:

- Writing Python program using Queues

Writing a Python program that will implement Queues operations

II. Methods

Instruction: Type the python codes below in your Colab. Reconstruct them by implementing Queues (FIFO) algorithm. Hint: You may use Array or Linked List

```
# Stack implementation in python

# Creating a stack
def create_stack():
    stack = []
    return stack
```

```
# Creating an empty stack
def is_empty(stack):
    return len(stack) == 0

# Adding items into the stack
def push(stack, item):
    stack.append(item)
    print("Pushed Element: " + item)

# Removing an element from the stack
def pop(stack):
    if (is  empty(stack)):
        return "The  stack is empty"
    return stack.pop()

stack = create  stack()
push(stack, str(1))
push(stack, str(2))
push(stack, str(3))
push(stack, str(4))
push(stack, str(5))

print("The  elements  in the stack are:"+ str(stack))
```

Answer the following questions:

1. What is the main difference between the stack and queue implementations in terms of element removal?
2. What would happen if we try to dequeue from an empty queue, and how is this handled in the code?
3. If we modify the enqueue operation to add elements at the beginning instead of the end, how would that change the queue behavior?
4. What are the advantages and disadvantages of implementing a queue using linked lists versus arrays?
5. In real-world applications, what are some practical use cases where queues are preferred over stacks?

III. Results

1. The main difference between a stack and queue is that stack follows the LIFO method (Last In, First out); the first element to be removed is the last element you input. On the other hand, Queue follows the FIFO method (first in, first out), meaning the first element you input is also the first element you will remove.
2. If you try to dequeue from an empty element, there is no element to remove, which can also cause IndexError.
3. If we enqueue at the beginning, the program would no longer behave like a queue. Instead, it will act like a stack, which follows the last in, first out.

4. A linked list implementation allows the queue to have a dynamic size, meaning the elements can be added or removed easily without worrying about fixed capacity. On the other hand, Array based implementation is simpler and allows direct access to elements using indexes.

5. There a lot of applications using the queues like printing, customer service, task scheduling, etc. This is because it follows the FIFO method.

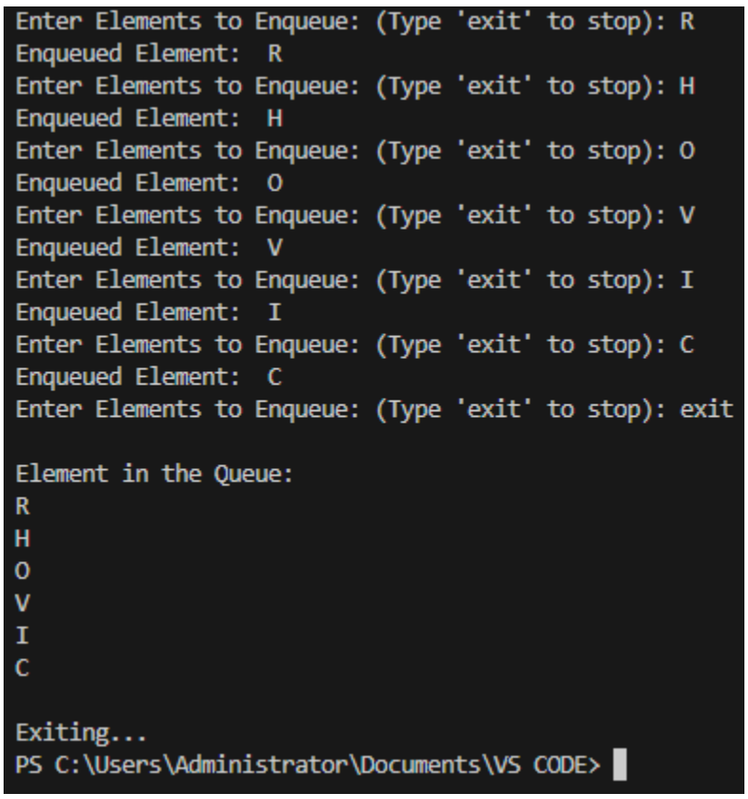

Figure 1.0 Program Output

IV. Conclusion

In conclusion, queues are important linear data structures, as it is follows the FIFO method, removing the most recent element in the data, allowing proper handling and management of the data. Both linked list and arrays can implement using a queue, but the choice between them depends on the specific needs of the program.

References

[1] Co Arthur O.. “University of Caloocan City Computer Engineering Department Honor Code,” UCC-CpE Departmental Policies, 2020.