Vietnam National University,
Ho Chi Minh City

University of Science

Faculty of Information Technology

Implementing Stack and Queue from scratch

CSC10004 – Data Structures and Algorithms

Ngo Nguyen The Khoa – 23127065 – 23CLC09

June 3, 2024

Contents

1 Result Screenshots

1.1 Stack (Array version)

'push' operation

```
-----------
| Stack max size: 3
| Stack size: 2
| Stack elements: 1 2

-----------
>>> Stack Lib
1. Push
2. Pop
0. Exit

=: Enter your choice: 1

=: Enter value to be pushed: 3

?: Continue: Yes(1) - No(0)
```

(a) Normal test cases

```
-----------
| Stack max size: 3
| Stack size: 3
| Stack elements: 1 2 3

-----------
>>> Stack Lib
1. Push
2. Pop
0. Exit

=: Enter your choice: 1

=: Enter value to be pushed: 4
!: Stack is full! Please pop before pushing

?: Continue: Yes(1) - No(0)
```

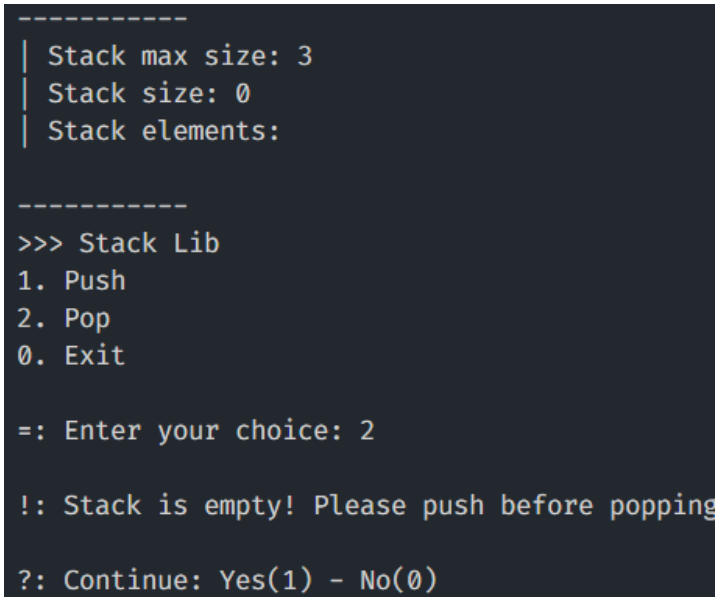
(b) When the stack is full

```
StackArray
❯• cpp main
-----------
| Stack is not initialized.
-----------
>>> Stack Lib
1. Push
2. Pop
0. Exit

=: Enter your choice: 1

!: Stack is not initialized.
?: Do you want to initialize the stack? Yes(1) - No(0)
1

=: Enter stack max size: 3

=: Enter value to be pushed: 1

?: Continue: Yes(1) - No(0)
```

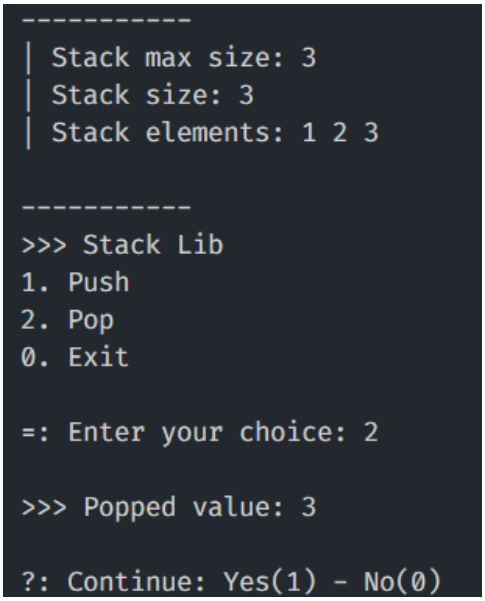
(c) When the stack is uninitialized

Figure 1: Screenshots of Stack (Array version) 'push' operation.

'pop' operation



(a) When the stack is empty

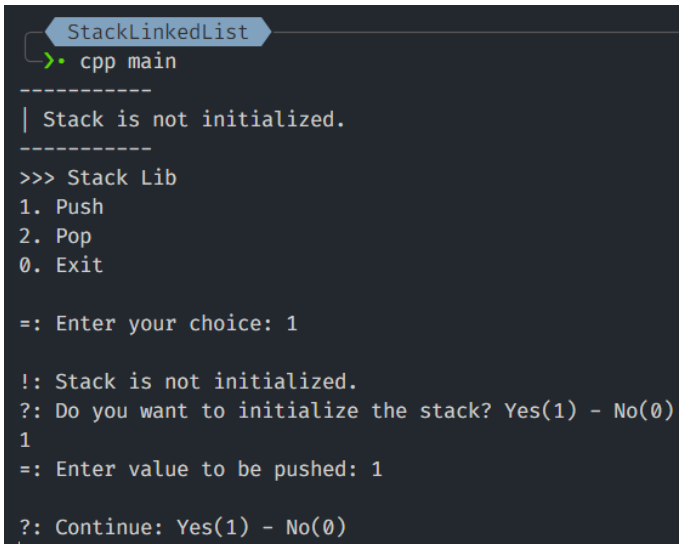


(b) Normal test cases

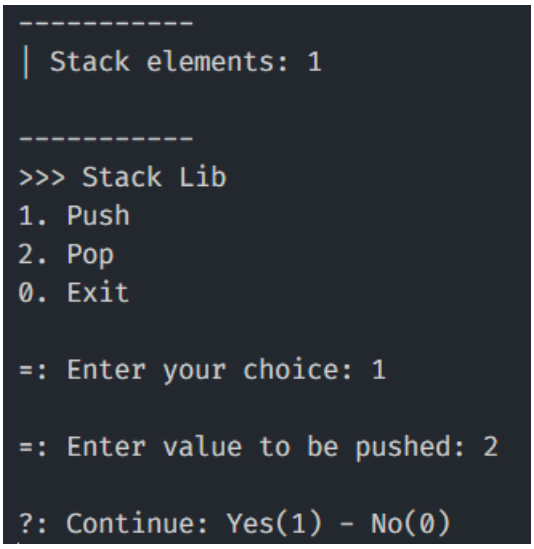
Figure 2: Screenshots of Stack (Array version) 'pop' operation.

1.2 Stack (Linked List version)

'push' operation



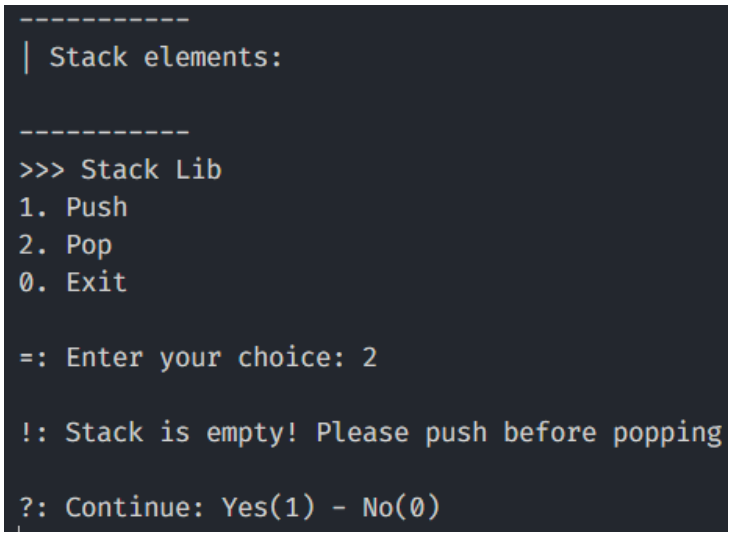
(a) When the stack is uninitialized



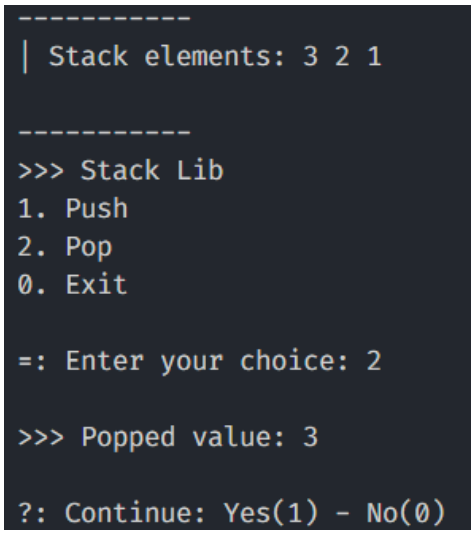
(b) Normal test cases

Figure 3: Screenshots of Stack (Linked List version) 'push' operation.

'pop' operation



(a) When the stack is empty



(b) Normal test cases

Figure 4: Screenshots of Stack (Linked List version) 'pop' operation.

1.3 Queue (Array version)

'enqueue' operation

```
-----------
| Queue max size: 3
| Queue size: 1
| Queue elements: 1
-----------
>>> Queue Lib
1. Enqueue
2. Dequeue
0. Exit

=: Enter your choice: 1

=: Enter value to be pushed: 2

?: Continue: Yes(1) - No(0)
```

(a) Normal test cases

```
-----------
| Queue max size: 3
| Queue size: 3
| Queue elements: 1 2 3
-----------
>>> Queue Lib
1. Enqueue
2. Dequeue
0. Exit

=: Enter your choice: 1

=: Enter value to be pushed: 4
!: Queue is full! Please dequeue before enqueueing

?: Continue: Yes(1) - No(0)
```

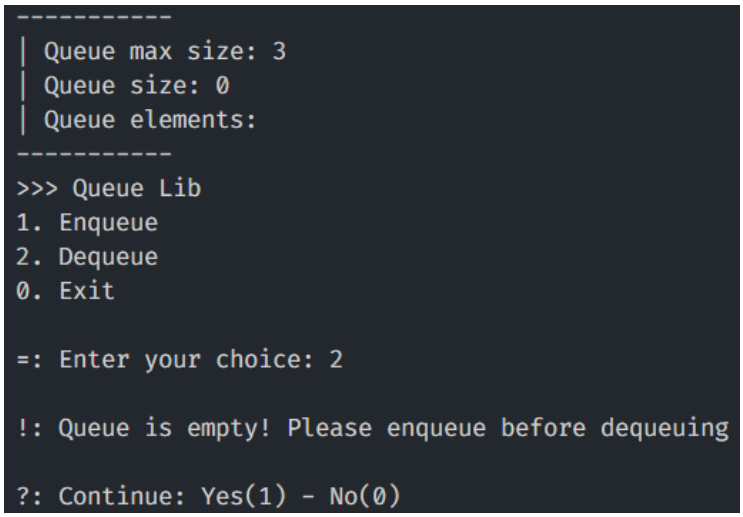
(b) When the queue is full

```
QueueArray
❯• cpp main
-----------
| Queue is not initialized.
-----------
>>> Queue Lib
1. Enqueue
2. Dequeue
0. Exit

=: Enter your choice: 1

!: Queue is not initialized.
?: Do you want to initialize the queue? Yes(1) - No(0)
1

=: Enter queue max size: 3

=: Enter value to be pushed: 1

?: Continue: Yes(1) - No(0)
```

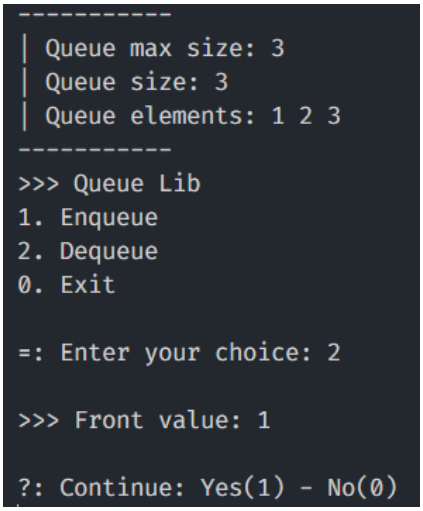
(c) When the queue is uninitialized

Figure 5: Screenshots of queue (Array version) 'enqueue' operation.

'dequeue' operation



(a) When the queue is empty

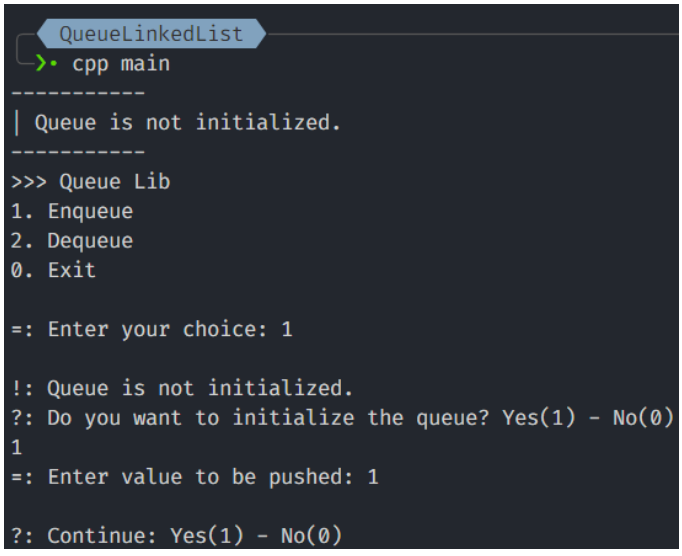


(b) Normal test cases

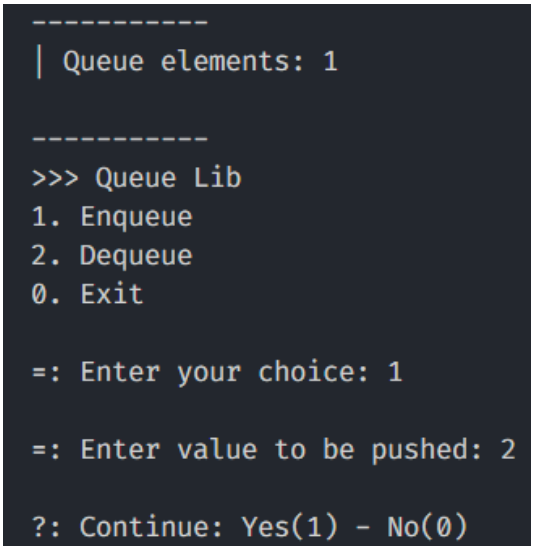
Figure 6: Screenshots of queue (Array version) 'dequeue' operation.

1.4 Queue (Linked List version)

'enqueue' operation



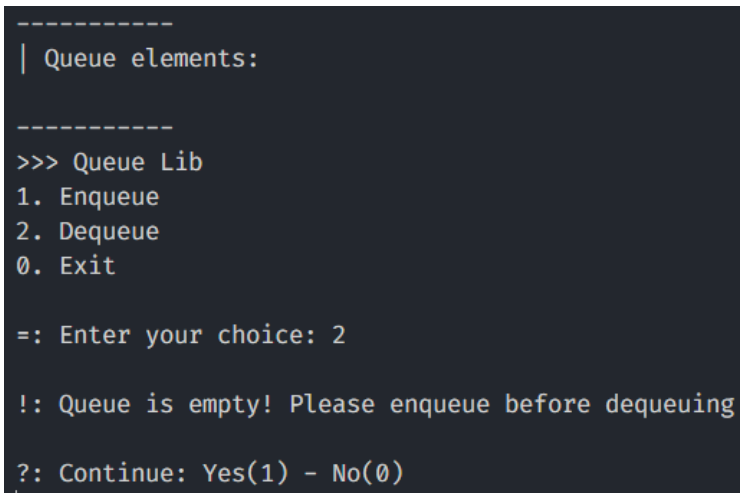
(a) When the queue is uninitialized



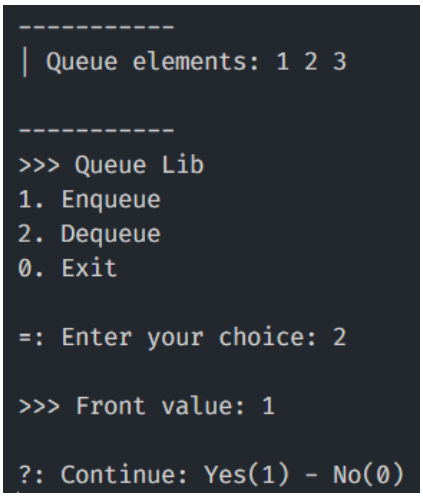
(b) Normal test cases

Figure 7: Screenshots of queue (Linked List version) 'enqueue' operation.

'dequeue' operation



(a) When the queue is empty



(b) Normal test cases

Figure 8: Screenshots of queue (Linked List version) 'dequeue' operation.

2 Recursive versions

2.1 Stack

Recursion Stack (Array version)

- **Theoretical Time Complexity:** Both implementations have the same theoretical time complexity of O(1) for `push` and O(n) for `copy` operations.
- **Practical Performance:**
  - Loop-based implementations are generally faster in practice because they avoid the overhead of function calls and recursion.
  - The recursive version doesn't work well with large data sets because of stack overflow.

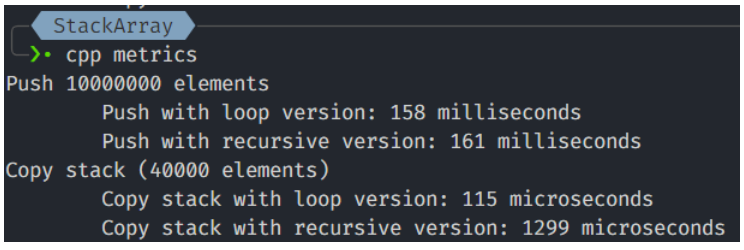


Figure 9: Recursive and loop version comparison of Stack (Array version)

Recursion Stack (Linked List version)

- **Theoretical Time Complexity:** Both implementations have the same theoretical time complexity of O(1) for `push`, O(n) for `copy` and `release` operations.
- **Practical Performance:**

- Loop-based implementations are generally faster in practice because they avoid the overhead of function calls and recursion.
- The recursive version doesn't work well with large data sets because of stack overflow.

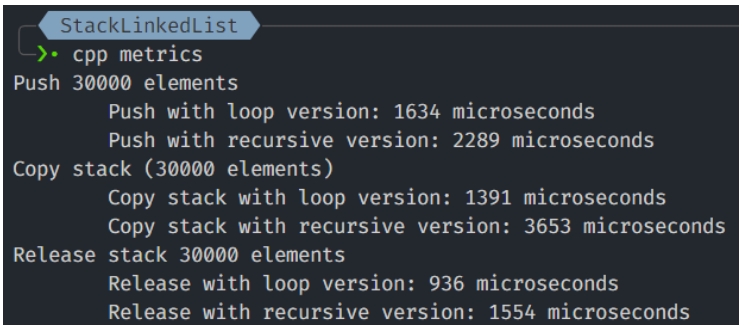


Figure 10: Recursive and loop version comparison of Stack (Linked List version)

2.2 Queue

Recursion Queue (Array version)

- **Theoretical Time Complexity:** Both implementations have the same theoretical time complexity of O(1) for `enqueue` and O(n) for `copy` operations.
- **Practical Performance:**
  - Loop-based implementations are generally faster in practice because they avoid the overhead of function calls and recursion.
  - The recursive version doesn't work well with large data sets because of stack overflow.

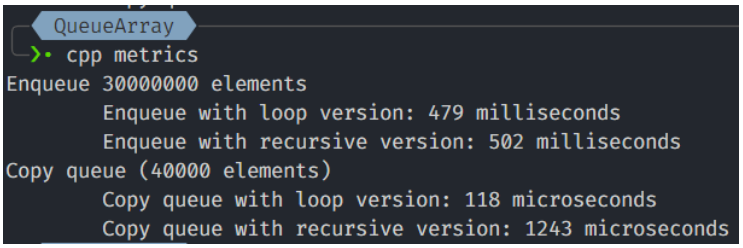


Figure 11: Recursive and loop version comparison of Queue (Array version)

Recursion Queue (Linked List version)

- **Theoretical Time Complexity:** Both implementations have the same theoretical time complexity of O(1) for `enqueue` and O(n) for `release` operations.
- **Practical Performance:**
  - Loop-based implementations are generally faster in practice because they avoid the overhead of function calls and recursion.

– The recursive version doesn't work well with large data sets because of stack overflow.

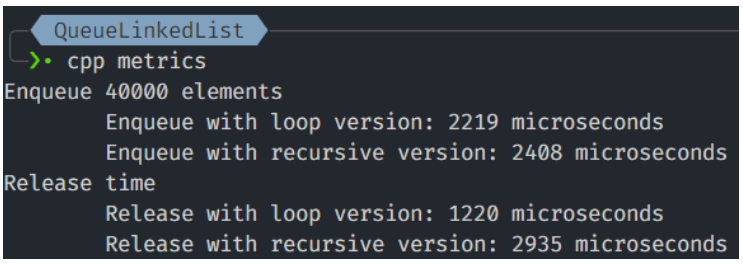


Figure 12: Recursive and loop version comparison of Queue (Linked List version)

3 Self-evaluation

No.	Details	Score
1	Stack (Array version)	100%
2	Stack (Linked List version)	100%
3	Queue (Array version)	100%
4	Queue (Linked List version)	100%
5	Recursive versions	100%
6	Report	100%

4 Exercise Feedback

4.1 What I learned

Because almost the things could be done easily, I have learned nothing new from this exercise.

4.2 What I found challenging

This exercise is quite simple and easy, so I have no difficulties to finish this task.

4.3 What I have used in this exercise

- I used C++ to implement the data structures and the test cases.
- I also used LaTeX to write this report.
- All the source code and the report are available in my github repo