Vietnam National University,
Ho Chi Minh City

UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

CSC10004 - Data Structure And Algorithms

EXERCISE I REPORT

Implementing Stack and Queue from scratch

Student Name	Student ID
Bui Minh Duy	23127040

Lecturer in charge:

Bui Duy Dang

Nguyen Thanh Tinh

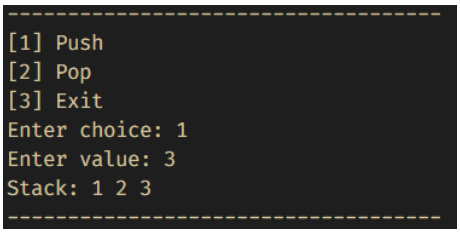
16/06/2024

Contents

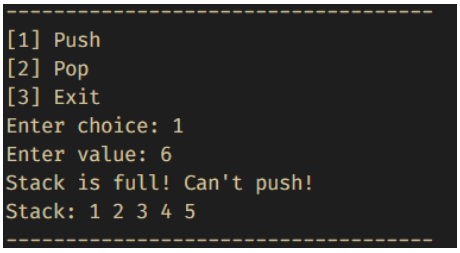
1 Experiment Details

1.1 Stack Array

Push Operation



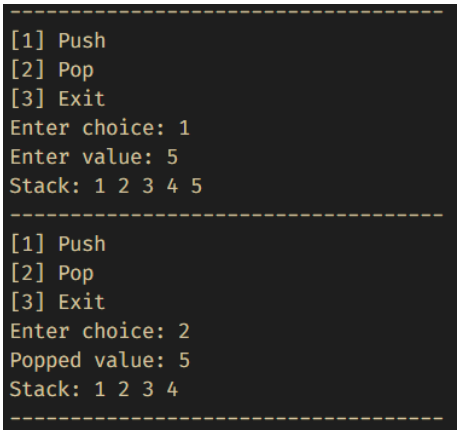
(a) Normal case



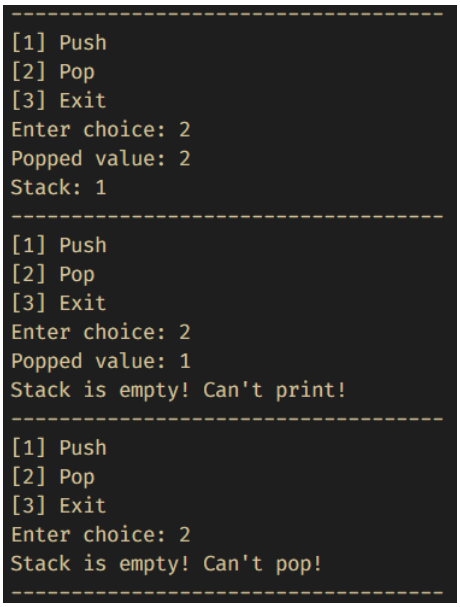
(b) Full case

Figure 1: Push elements in Stack Array of size 5

Pop Operation



(a) Normal case

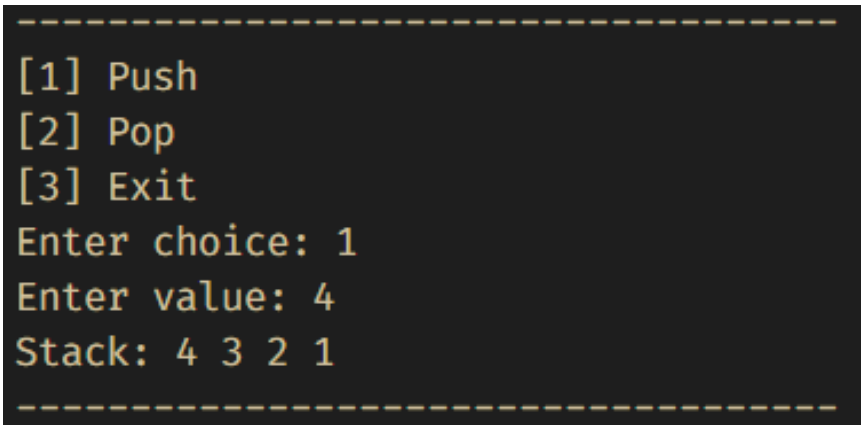


(b) Empty case

Figure 2: Pop elements in Stack Array of size 5

1.2 Stack Linked List

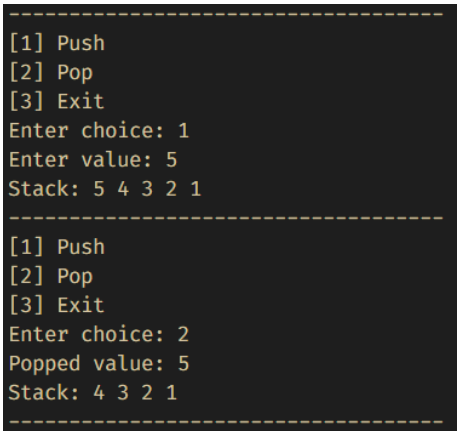
Push Operation



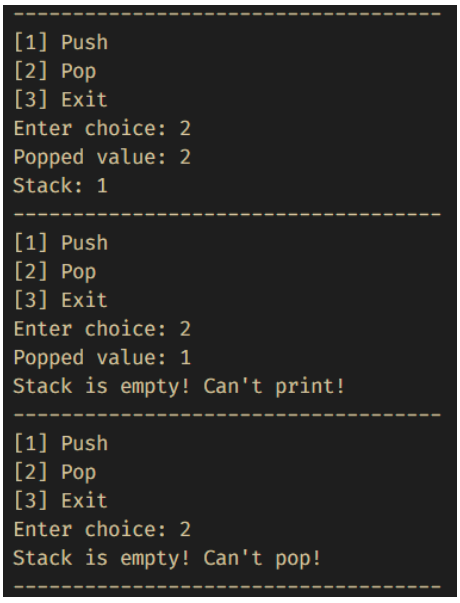
(a) Normal Case

Figure 3: Push elements in Stack Linked List

Pop Operation



(a) Normal case



(b) Empty case

Figure 4: Pop elements in Stack Linked List

1.3 Queue Array

Enqueue Operation

```
-----------------------------------
[1]. Enqueue
[2]. Dequeue
[3]. Exit
Enter choice: 1
Enter value: 3
Queue: 1 2 3
-----------------------------------
```

(a) Normal case

```
-----------------------------------
[1]. Enqueue
[2]. Dequeue
[3]. Exit
Enter choice: 1
Enter value: 6
Queue is full! Can't enqueue!
Queue: 1 2 3 4 5
-----------------------------------
```

(b) Full case

Figure 5: Enqueue elements in Queue Array of size 5

Dequeue Operation

```
-----------------------------------
[1]. Enqueue
[2]. Dequeue
[3]. Exit
Enter choice: 1
Enter value: 4
Queue: 1 2 3 4
-----------------------------------
[1]. Enqueue
[2]. Dequeue
[3]. Exit
Enter choice: 2
Dequeued item: 1
Queue: 2 3 4
-----------------------------------
```

(a) Normal case

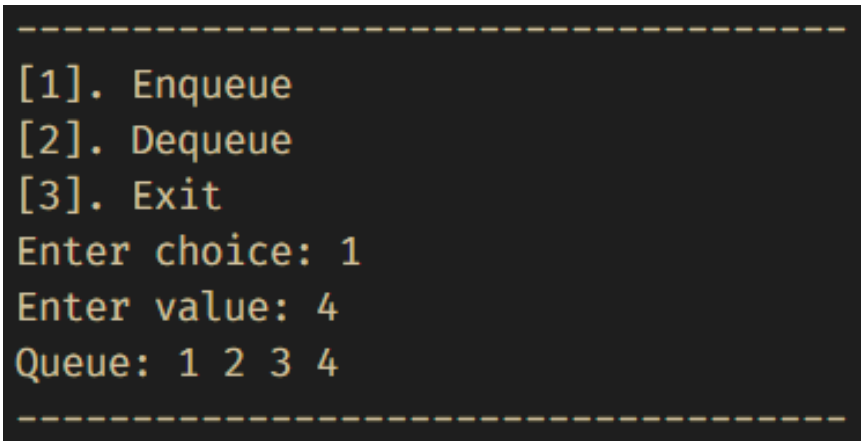
```
-----------------------------------
[1]. Enqueue
[2]. Dequeue
[3]. Exit
Enter choice: 2
Dequeued item: 3
Queue: 4
-----------------------------------
[1]. Enqueue
[2]. Dequeue
[3]. Exit
Enter choice: 2
Dequeued item: 4
Queue is empty! Can't print!
-----------------------------------
[1]. Enqueue
[2]. Dequeue
[3]. Exit
Enter choice: 2
Queue is empty! Can't dequeue.
-----------------------------------
```

(b) Empty case

Figure 6: Dequeue elements in Queue Array of size 5

1.4 Queue Linked List

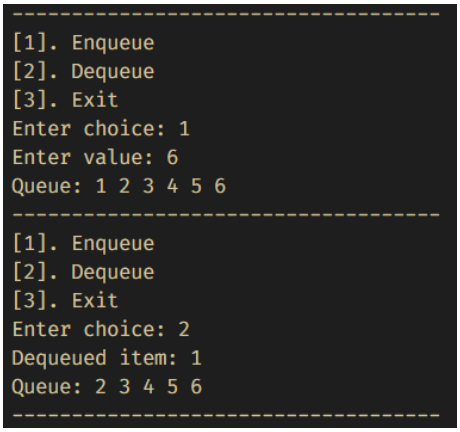
Enqueue Operation



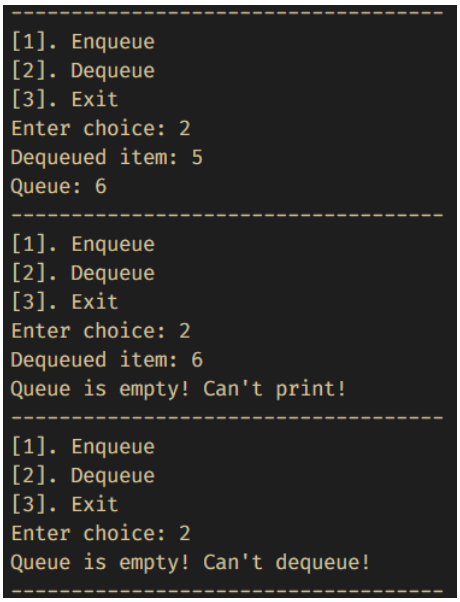
(a) Normal case

Figure 7: Enqueue elements in Queue Linked List

Dequeue Operation



(a) Normal case



(b) Empty case

Figure 8: Dequeue elements in Queue Linked List

2 Recursive versions

2.1 Stack

Recursive Stack Array

- **Theoretical Time Complexity:** Both implementations have the same theoretical time complexity of O(n) for `copy` operations.
- **Actual execution time:**

```
Time taken to copy stack of 10000 elements:
Loop version: 15 microseconds
Recursive version: 41 microseconds
```

Figure 9: Comparison between Loop version and Recursive version

Recursive Stack Linked List

- **Theoretical Time Complexity:** Both implementations have the same theoretical time complexity of O(n) for `copy` and `release` operations.
- **Actual execution time:**

```
Time taken to copy stack of 10000 elements:
Loop version: 289 microseconds
Recursive version: 344 microseconds
```

Figure 10: Comparison between Loop version and Recursive version

```
Time taken to release stack of 10000 elements:
Loop version: 147 microseconds
Recursive version: 163 microseconds
```

Figure 11: Comparison between Loop version and Recursive version

2.2 Queue

Recursive Queue Array

- **Theoretical Time Complexity:** Both implementations have the same theoretical time complexity of O(n) for `copy` operations.
- **Actual execution time:**

```
Time taken to copy a queue of 10000 elements:
Loop version: 29 microseconds
Recursive version: 80 microseconds
```

Figure 12: Comparison between Loop version and Recursive version

Recursive Queue Linked List

- **Theoretical Time Complexity:** Both implementations have the same theoretical time complexity of O(n) for `copy` and `release` operations.
- **Actual execution time:**

```
Time taken to copy a queue of 10000 elements:
Loop version: 287 microseconds
Recursive version: 335 microseconds
```

Figure 13: Comparison between Loop version and Recursive version

```
Time taken to release a queue of 10000 elements:
Loop version: 145 microseconds
Recursive version: 185 microseconds
```

Figure 14: Comparison between Loop version and Recursive version

2.3 Conclusion

- Though Loop version and Recursive Version have the same theoretical time complexity of O(n), the actual execution time shows that Loop version is faster in all cases.
- When working with large data sets, the Loop version still executes as normal, but the Recursive version might encounter Stack Overflow.

3 Self-evaluation

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

4 Exercise Feedback

4.1 What I learned

- Better understanding about the internal processing of stack and queue.
- How to use function templates in C++.

4.2 What was difficult

- The approach of using recursion is quite challenging, it took a significant amount of time and effort to complete the set-up.