

EECS2040 Data Structure Hw #2 (Chapter 3 Stack/Queue)

due date 4/23/2021

by 107061123, 孫元駿

Part 2 Coding (50%)

You should submit:

- (a) All your source codes (C++ file).
- (b) Show the execution trace of your program.

1. (30%) Based on the circular queue and template queue ADT in **ADT 3.2** in textbook (or pptx pp.79), write a C++ program to implement the queue ADT.

Then add two more functions to

- (a) Return the size and capacity of a queue.
- (b) Merge two queues into a one by alternately taking elements from each queue. The relative order of queue elements is unchanged. What is the complexity of your function?

You should **demonstrate all the functions** using at least one example.

Ans:

In ADT 3.2 mention 1 Constructor and 5 functions, and I write three more functions.

Functions in ADT 3.2: *IsEmpty, Front, Rear, Push, Pop*

Functions addition: *Capacity, Size, MergeQ(for merging two queue)*

Command Table:

isempty -> IsEmpty / front -> Front / rear -> Rear / push <element> -> Push

pop -> Pop / capacity -> Capacity / size -> Size

do it after finish inserting two queue -> MergeQ

```
11th queue
Set the queue capacity as: 10
Command: push <element> / pop / front / rear / size / capacity / end
queue is empty.
size
The size is 0
capacity
The capacity is 10
push a
push b
push c
front
The front is a
rear
The rear is c
isempty
queue is not empty.
size
The size is 3
capacity
The capacity is 10
pop
Pop out a
front
The front is b
end
21th queue
Set the queue capacity as: 2
Command: push <element> / pop / front / rear / size / capacity / end
queue is empty.
size
The size is 0
capacity
The capacity is 2
push 1
push 2
push 3
push 4
Wrong command.
push 4
push 5
front
The front is 1
rear
The rear is 5
isempty
queue is not empty.
size
The size is 5
capacity
The capacity is 8
pop
Pop out 1
pop
Pop out 2
front
The front is 3
end
Merge two queue
The 0 Queue (first to last) -> b c
The 1 Queue (first to last) -> 3 4 5
Merge size is 5
1: b
2: 3
3: c
4: 4
5: 5
```

About my program:

You are going to create two queues and the program will merge them together.

First, you need to insert an integer as the queue capacity. Then the program will create first queue container. After that you can insert command as the *command table* above. You can finish pushing and popping the element by command “end”.

Then, you can create the second queue and repeat the step above.

In the end, the program will print out the left element respectively and also merge the two queue together and print out the queue after merging.

2. (35%) Referring to **Program 3.13** in textbook (pptx pp.94),

(a) Implement Stack as a publicly derived class of Bag using template.

Demonstrate your C++ code using at least two element types (e.g., int, float,...). **Show results** of a series of Pushes and Pops and Size functions.

(b) Implement Queue as a publicly derived class of Bag using template.

Demonstrate your C++ code using at least two element types (e.g., int, float,...). **Show results** of a series of Pushes and Pops and Size functions.

(c) A template double-ended queue (deque) is a linear list in which additions and deletions may be made at either end. Implement the class Deque as a publicly derived templated class of Queue. The class Deque must have public functions (either via inheritance from Queue or by direct implementation in Deque) to add and delete elements from either end of the deque and also to return an element from either end. The complexity of each function (excluding array doubling) should be $\Theta(1)$.

Demonstrate your C++ code using at least two element types (e.g., int, float,...). **Show results** of a series of two types of Pushes and Pops and Size functions to illustrate your code is working.

Ans:

(a)

Three functions to show, *Pushes*, *Pops*, *Size*

Command Table:

push <element> -> *Pushes*

pop -> *Pops*

size -> *Size*

Type 1: char

```
~/Desktop/data_structure/hw2/hw2_2  P master  g++ -Wall -std=c++17 hw2_2_stack.cpp 22:31:01
~/Desktop/data_structure/hw2/hw2_2  P master  ./a.out 22:31:03

This is a Stack program.
Set the initial capacity as: 10
Command: push <element> / pop / size / quit
>> push 5
>> push T
>> push A
>> push C
>> push K
>> push ?
>> push !
>> size
size is 7
>> pop
pop out !
>> pop
pop out ?
>> size
size is 5
>> quit
The left element (first to last): S T A C K
~/Desktop/data_structure/hw2/hw2_2  P master  22:32:01
```

Type 2: float

```
~/Desktop/data_structure/hw2/hw2_2  P master  g++ -Wall -std=c++17 hw2_2_stack.cpp 22:32:01
~/Desktop/data_structure/hw2/hw2_2  P master  ./a.out 22:34:40

This is a Stack program.
Set the initial capacity as: 10
Command: push <element> / pop / size / quit
>> push 0.1
>> push 1.1
>> push 1.2
>> push 1.3
>> size
size is 4
>> pop
pop out 1.3
>> size
size is 3
>> quit
The left element (first to last): 0.1 1.1 1.2
~/Desktop/data_structure/hw2/hw2_2  P master  22:35:18
```

(b)

Three functions to show, *Pushes*, *Pops*, *Size*

Command Table:

push <element> -> *Pushes*

pop -> *Pops*

size -> *Size*

Type 1: char

```
~/Desktop/data_structure/hw2/hw2_2  P master  g++ -Wall -std=c++17 hw2_2_queue.cpp  22:35:18
~/Desktop/data_structure/hw2/hw2_2  P master  ./a.out  22:36:46

This is a Queue program.
Set the initial capacity as: 10
Command: push <element> / pop / size / show / quit
>> push ?
>> push !
>> push Q
>> push U
>> push E
>> push U
>> push E
>> size
>> size
size is 7
>> pop
pop out ?
>> pop
pop out !
>> size
size is 5
>> quit
The left element (first to last): Q U E U E
~/Desktop/data_structure/hw2/hw2_2  P master  22:37:48
```

Type 2: float

```
~/Desktop/data_structure/hw2/hw2_2  P master  g++ -Wall -std=c++17 hw2_2_queue.cpp  22:37:48
~/Desktop/data_structure/hw2/hw2_2  P master  ./a.out  22:39:25

This is a Queue program.
Set the initial capacity as: 10
Command: push <element> / pop / size / show / quit
>> push 0.1
>> push 1.1
>> push 2.1
>> push 3.1
>> size
size is 4
>> pop
pop out 0.1
>> size
size is 3
>> quit
The left element (first to last): 1.1 2.1 3.1
~/Desktop/data_structure/hw2/hw2_2  P master  22:48:03
```

(c)

Five functions to show, *Pushes(Stack)*, *Pushes(Queue)*, *Pop(Stack)*, *Pop(Queue)*, *Size*
Command Table:

push_top <element> -> Pushes(Stack)

push_bottom <element> -> Pushes(Queue)

pop_top -> Pops(Stack)

pop_bottom -> Pops(Queue)

size -> Size

Type 1: char

```
~/Desktop/data_structure/hw2/hw2_2  P master  g++ -Wall -std=c++17 hw2_2_deque.cpp  22:41:52
~/Desktop/data_structure/hw2/hw2_2  P master  ./a.out  22:41:57

This is a Deque program.
Set the initial capacity as: 10
Command: push_top <element> / push_bottom <element> / pop_top / pop_bottom / size / show / quit
>> push_top D
>> push_top E
>> push_top Q
>> push_top U
>> push_top E
>> size
size is 5
>> push_bottom *
>> push_bottom *
>> push_top ?
>> push_top ?
>> pop_bottom
pop out *
>> pop_top
pop out ?
>> size
size is 7
>> quit
The left element (first to last): * D E Q U E ?
~/Desktop/data_structure/hw2/hw2_2  P master  22:44:15
```

Type 2: float

```
~/Desktop/data_structure/hw2/hw2_2  P master  g++ -Wall -std=c++17 hw2_2_deque.cpp  22:46:17
~/Desktop/data_structure/hw2/hw2_2  P master  ./a.out  22:46:41

This is a Deque program.
Set the initial capacity as: 10
Command: push_top <element> / push_bottom <element> / pop_top / pop_bottom / size / show / quit
>> push_top 1.1
>> push_top 1.11
>> push_top 1.12
>> push_top 1.13
>> push_bottom 1.09
>> push_bottom 1.08
>> push_bottom 1.07
>> pop_top
pop out 1.13
>> pop_bottom
pop out 1.07
>> size
size is 5
>> quit
The left element (first to last): 1.08 1.09 1.1 1.11 1.12
~/Desktop/data_structure/hw2/hw2_2  P master  22:48:20
```

Then, the program will print out the stack/queue/deque.

- (a) Demonstrate your maze program using the maze shown in **Figure 3.11** in textbook.

```

~/Desktop/data_structure/hw2/hw2.3  P master 00:48:38
~/Desktop/data_structure/hw2/hw2.3  P master 00:48:37

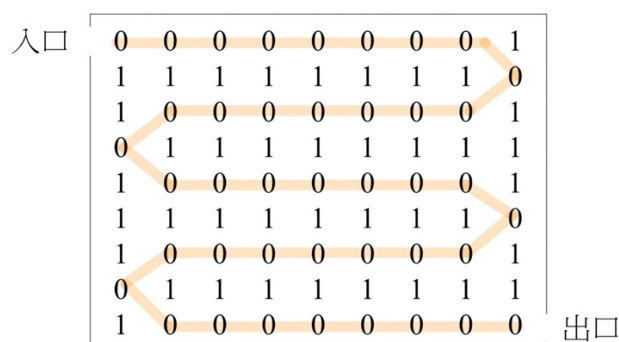
0
1
1 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1
1 0 0 0 1 1 0 1 1 1 0 0 1 1 1 1
0 1 1 0 0 0 0 1 1 1 1 0 0 1 1
1 1 0 1 1 1 0 1 1 0 1 1 0 0
1 1 0 1 0 0 1 0 1 1 1 1 1 1 1
0 0 1 1 0 1 1 1 0 1 0 0 1 0 1
0 0 1 1 0 1 1 1 0 1 0 0 1 0 1
0 1 1 1 1 0 0 1 1 1 1 1 1 1
0 0 1 1 0 1 1 0 1 1 1 1 1 0 1
1 1 0 0 0 1 1 0 1 1 0 0 0 0
0 0 1 1 1 1 0 0 0 1 1 1 1
0 1 0 0 1 1 1 1 0 1 1 1 0 E
-----

Total steps = 29
0: (1,1) go SE to
1: (2,2) go NE to
2: (1,3) go E to
3: (1,4) go E to
4: (1,5) go SW to
5: (2,4) go SE to
6: (3,5) go W to
7: (3,4) go SW to
8: (4,3) go S to
9: (5,3) go SW to
10: (6,2) go S to
11: (7,2) go SE to
12: (8,1) go SE to
13: (9,2) go SE to
14: (10,3) go E to
15: (10,4) go NE to
16: (9,5) go NE to
17: (8,6) go E to
18: (8,7) go SE to
19: (9,8) go S to
20: (10,8) go SE to
21: (11,9) go E to
22: (11,10) go NE to
23: (10,11) go E to
24: (10,12) go E to
25: (10,13) go NE to
26: (9,14) go SE to
27: (10,15) go S to
28: (11,15) go S to
The end (12,15)

~/Desktop/data_structure/hw2/hw2.3  P master 00:48:50

```

- (b) Find a path through the maze shown **Figure 3.14** in textbook



- (c) Trace out the action of function path (**Program 3.16**) on the maze shown.
Compare this to your own attempt in (b).

```

~/Desktop/data_structure/ha2/ha2_3  P master  g++ -Wall -std=c++17 Path.cpp
~/Desktop/data_structure/ha2/ha2_3  P master  ./a.out
- - - - -
5 0 0 0 0 0 0 0 1 -
- 1 1 1 1 1 1 1 0 -
- 0 0 0 0 0 0 0 1 -
- 0 1 1 1 1 1 1 1 -
- 1 0 0 0 0 0 0 1 -
- 1 1 1 1 1 1 0 -
- 1 0 0 0 0 0 0 1 -
- 0 1 1 1 1 1 1 -
- 1 0 0 0 0 0 0 E
- - - - -
Total steps = 49
0: (1,1) go E to
1: (1,2) go E to
2: (1,3) go E to
3: (1,4) go E to
4: (1,5) go E to
5: (1,6) go E to
6: (1,7) go E to
7: (1,8) go SE to
8: (2,9) go SW to
9: (3,8) go W to
10: (3,7) go W to
11: (3,6) go W to
12: (3,5) go W to
13: (3,4) go W to
14: (3,3) go W to
15: (3,2) go SW to
16: (4,1) go SE to
17: (5,2) go E to
18: (5,3) go E to
19: (5,4) go E to
20: (5,5) go E to
21: (5,6) go E to
22: (5,7) go E to
23: (5,8) go SE to
24: (6,9) go SW to
25: (7,8) go W to
26: (7,7) go W to
27: (7,6) go W to
28: (7,5) go W to
29: (7,4) go W to
30: (7,3) go W to
31: (7,2) go SW to
32: (6,1) go SE to
33: (9,2) go E to
34: (9,3) go E to
35: (9,4) go E to
36: (9,5) go E to
37: (9,6) go E to
38: (9,7) go E to
39: (9,8) go E to
The end (9,9)
~/Desktop/data_structure/ha2/ha2_3  P master

```

About my program (a)(c):

You are going to get the path by (x,y) and direction every step to solve the maze.

First, you need to have a maze txt file and this cpp file under the same folder.

Then run the program, you will get the step and the path.

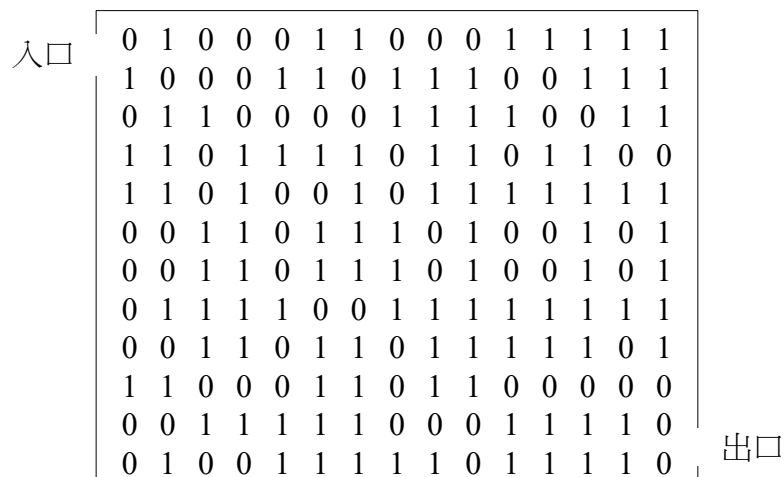


Figure 3.11: 一個迷宮的例子（你能找出一條路徑嗎？）

入口	0	0	0	0	0	0	0	0	1
	1	1	1	1	1	1	1	1	0
	1	0	0	0	0	0	0	0	1
	0	1	1	1	1	1	1	1	1
	1	0	0	0	0	0	0	0	1
	1	1	1	1	1	1	1	1	0
	1	0	0	0	0	0	0	0	1
	0	1	1	1	1	1	1	1	1
	1	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0
出口									

Figure 3.14: 唯一路徑的迷宮圖