Practice #5

Stacks and Queues

Yunmin Go

School of CSEE

HANDONG GLOBAL UNIVERSITY

Practice #5 TO-DO List

To-Do	Submission	Notes
Stack	Screenshot and source code (stack.cpp)	p.11-18, Chapter 3
Queue	Screenshot and source code (queue.cpp)	p.27-32, Chapter 3
Maze Algorithm	Screenshot and source code (maze.cpp)	p.44, Chapter 3

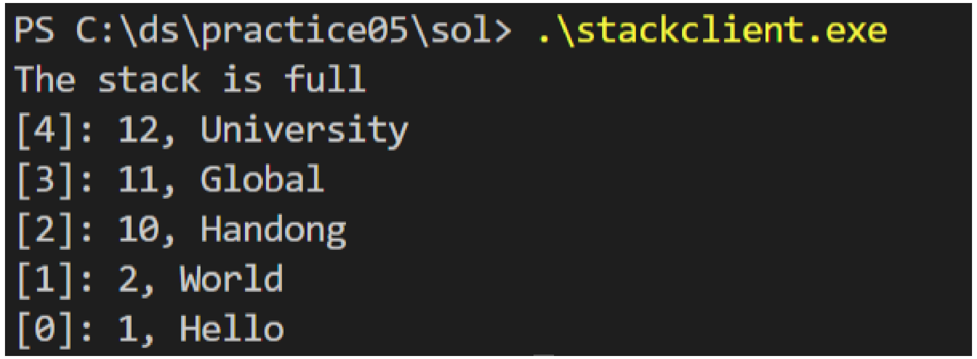
- Upload your screenshot and source codes on LMS by 11pm on 3/31 (Wed).
  - All your screenshots should be merged in one pdf file, screenshot.pdf.
  - Your pdf and all source codes should be compressed into zip file.
- File name: practice05_Your Student ID_Name.zip (only zip, not pdf, docx, c, etc)
  - ex) practice05_20400022_고윤민.zip

Stack

- Implement a stack with structure pointer
  - Skeleton code: stack.cpp (stackclient.cpp: no need to change)
  - Refer to p.11-18, Chapter 3.
  - Element is defined in structure type
  - Stack class has an array of ‘max_size’ elements
  - Complete member functions of Stack class
    - Stack(int num): create Element array of ‘max_size’
      - max_size is a parameter of constructor
    - IsFullS(), IsEmptyS(), Push(), Pop()

Stack

- Expected results



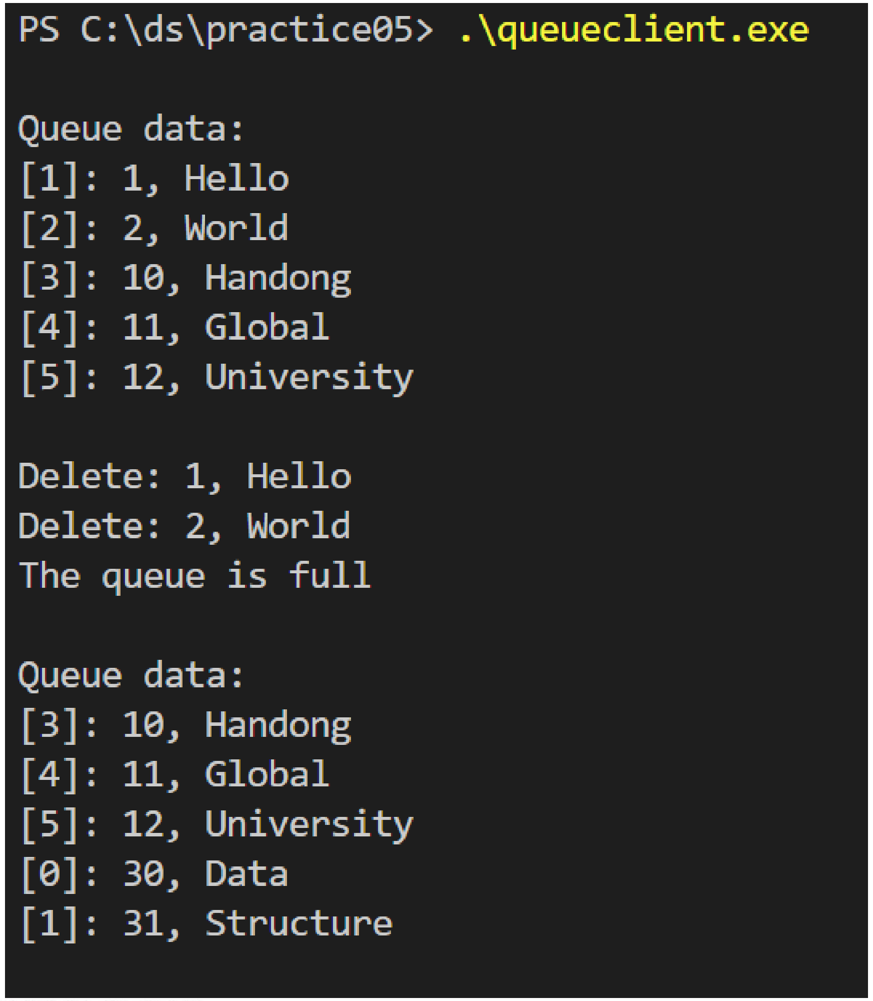
```
PS C:\ds\practice05\sol> .\stackclient.exe
The stack is full
[4]: 12, University
[3]: 11, Global
[2]: 10, Handong
[1]: 2, World
[0]: 1, Hello
```

Queue

- Implement a queue with class pointer
  - Skeleton code: queue.cpp (queueclient.cpp: no need to change)
  - Refer to p.27-32, Chapter 3.
  - Element is defined in class type
    - Check the constructor in Element class and the main function.
  - Queue class has an array of ‘max_size’ elements
  - Complete member functions of Stack class
    - Queue(int num): create Element array of ‘max_size’
      - max_size is a parameter of constructor
    - IsFullQ(), IsEmptyQ(), AddQ(), DeleteQ()
    - Print(): Print all of elements in queue (from front to rear)
      - Print format: refer to the next slide

Queue

- Expected results



```
PS C:\ds\practice05> .\queueclient.exe

Queue data:
[1]: 1, Hello
[2]: 2, World
[3]: 10, Handong
[4]: 11, Global
[5]: 12, University

Delete: 1, Hello
Delete: 2, World
The queue is full

Queue data:
[3]: 10, Handong
[4]: 11, Global
[5]: 12, University
[0]: 30, Data
[1]: 31, Structure
```

Maze Algorithm

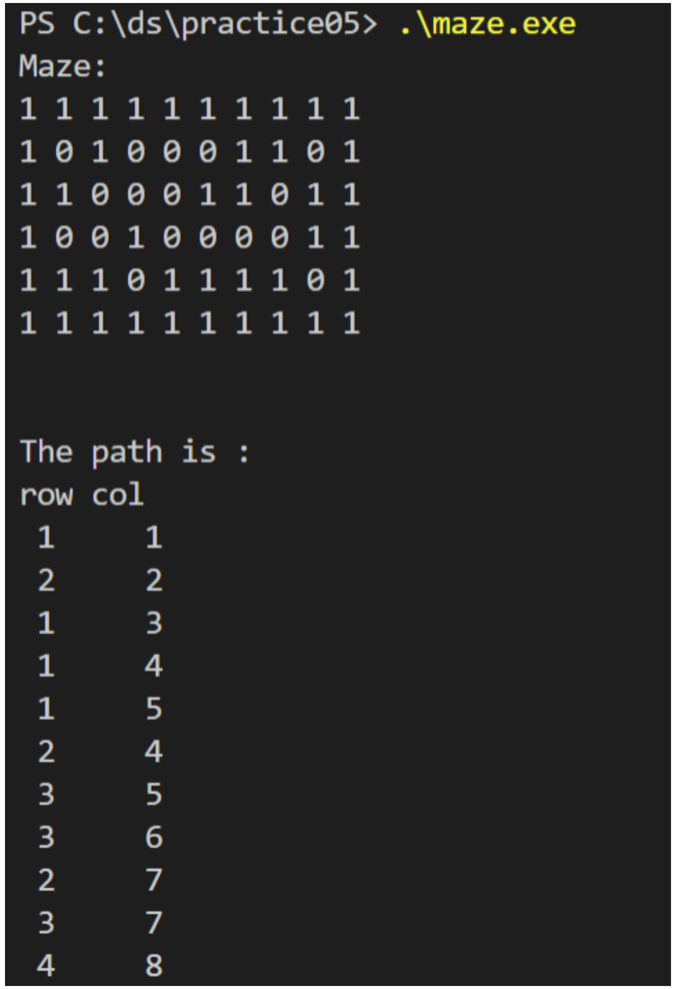
- Complete a maze algorithm
  - Skeleton code: maze.cpp
  - Refer to p.44, Chapter 3.
  - Element is defined in <u>structure</u> type
  - Use stack class (stack.cpp: first item)
  - Complete findPath() in Maze class

```
initialize a stack to the maze's entrance coordinates (1,1) and
direction to north (0)
while (stack is not empty) {
    <row,col,dir> = coordinate and direction from top of stack;
    while (dir < 8 AND Exit is not found) {
        <next_row, next_col> = coordinate of next move;
        if (<next_row, next_col> == < EXIT_ROW, EXIT_COL> )
            success;
        if (maze[next_row][next_col] == 0
            && mark[next_row][next_col] == 0))
        {
            mark[next_row][next_col] = 1;
            add <row, col, dir+1> to the top of the stack;
            row = next_row;
            col = next_col;
            dir = north;
        } else
            dir++;
    }
}
```

HANDONG GLOBAL UNIVERSITY

Maze Algorithm

- Expected results



```
PS C:\ds\practice05> .\maze.exe
Maze:
1 1 1 1 1 1 1 1 1 1
1 0 1 0 0 0 1 1 0 1
1 1 0 0 0 1 1 0 1 1
1 0 0 1 0 0 0 0 1 1
1 1 1 0 1 1 1 1 0 1
1 1 1 1 1 1 1 1 1 1


The path is :
row col
 1     1
 2     2
 1     3
 1     4
 1     5
 2     4
 3     5
 3     6
 2     7
 3     7
 4     8
```