Chap 3. Stacks and Queues

Stack

- ADT 3.1: abstract data type Stack
- Stack CreateS(maxStackSize) ::= // § 3.1
- Program 3.1: add an item to a stack
- Program 3.2: delete from a stack
- Program 3.3: stack full

Queue

- ADT 3.2: abstract data type Queue
- Queue CreateQ(maxQueueSize) ::= // § 3.3
- Program 3.5: add to a queue
- Program 3.6: delete from a queue
- Circular queue:
  - Initialization: front=rear=0
  - Program 3.7: add to a circular queue
  - Program 3.8: delete from a circular queue

Maze

- Figure 3.8: an example maze (can you find a path?)
  - Maze의 표현: 2D array maze[][]
- Figure 3.9: allowable moves
- Figure 3.10: table of moves
- Type offsets and array move[8] // § 3.5
- Legal move:
  - 2D array mark[][]
  - Condition: maze[i][j]=0 and mark[i][j]=0
- Type element // § 3.5
- Program 3.11: initial maze algorithm
- Program 3.12: maze search function

Evaluating postfix expressions

- Figure 3.13: infix and postfix notation
- Figure 3.14: postfix evaluation
- Program 3.13: function to evaluate a postfix expression
  - 열거형 상수 enum
  - ASCII code of '0'~'9': 30H~39H
- Program 3.14: function to get a token from the input string

Infix to postfix

- Figure 3.15: translation of a+b*c to postfix
- Figure 3.16: translation of a*(b+c)*d to postfix
- Figure 3.12: precedence hierarchy for C
- isp and icp arrays // § 3.6.3
- Program 3.15: function to convert from infix to postfix
  - printToken(token_type)
- § 3.6.3 Exercises 1번 (a),(d)

Multiple stacks and queues

- Declarations // § 3.7
- Figure 3.18: initial configuration for n stacks in memory[m]
- Program 3.16: add an item to the i-th stack
- Program 3.17: delete an item from the i-th stack
- Figure 3.19: configuration when stack i meets stack i+1 but memory is not full