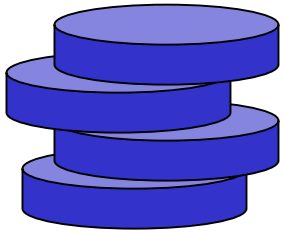
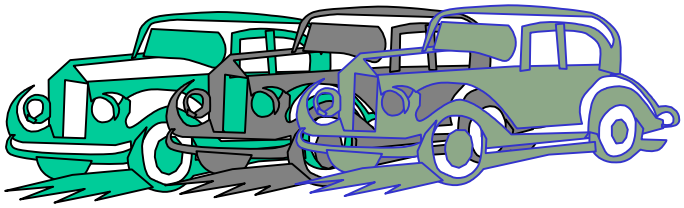




Lecture5: 스택(5.1절), 큐(5.2절), 데크(5.3절)

김 강 희
khkim@ssu.ac.kr

요약

- 스택 (stack)
  - 임의의 객체들을 삽입한 순서대로 저장하는 자료 구조
  - 삭제는 Last In, First Out 방식으로 수행됨
  - 최근성(recency)가 중요한 상황에서 많이 사용됨
    - 웹 페이지 방문 기록, 에디터의 undo, …
- 큐 (queue)
  - 임의의 객체들을 삽입한 순서대로 저장하는 자료 구조
  - 삭제는 First In, First Out 방식으로 수행됨
  - 시간 순서(time order)가 중요한 상황에서 많이 사용됨
    - 대기자 명단, 생산자-소비자 관계 (프린터), …
- 데크 (deque: double-ended queue)
  - 임의의 객체들을 큐 형태로 저장하되, 큐의 앞과 뒤에서 각기 삽입과 삭제가 가능한 자료 구조
  - 데크를 스택 또는 큐처럼 사용할 수 있음

5.1절 스택

```
template <typename E>
class Stack  {
public:
   int size() const;
   bool empty() const;
   const E& top() const throw(StackEmpty);
   void push(const E& e);
   void pop() throw(StackEmpty);
}
```

활용예: 산술식 계산기

Slide by Matt Stallmann included with permission.

14 – 3 * 2 + 7 = (14 – (3 * 2)) + 7

Operator precedence

* has precedence over +/–

Associativity

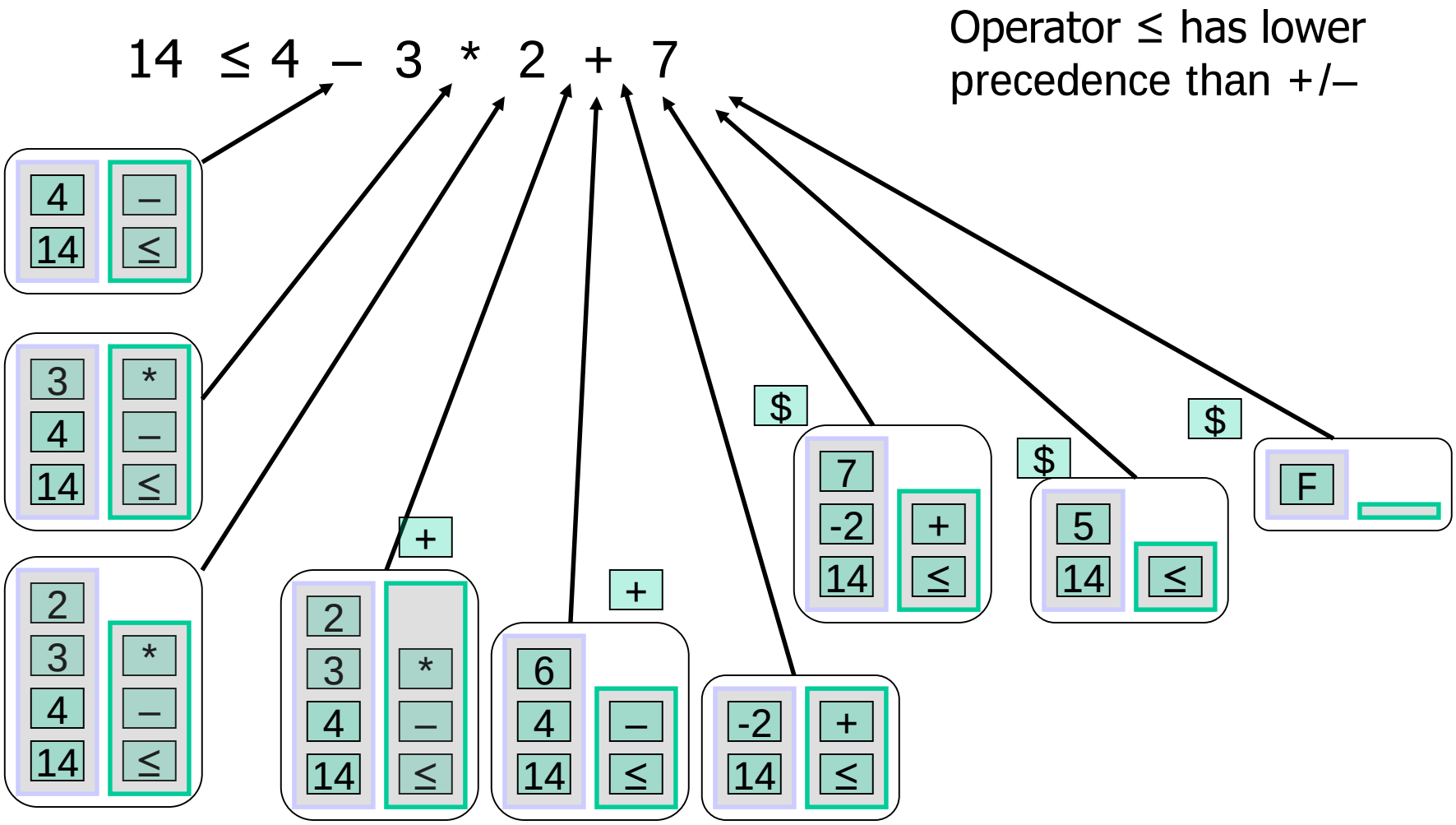
operators of the same precedence group evaluated from left to right

Example: (x – y) + z rather than x – (y + z)

Idea: push each operator on the stack, but first pop and perform higher and *equal* precedence operations.

활용예: 산술식 계산기

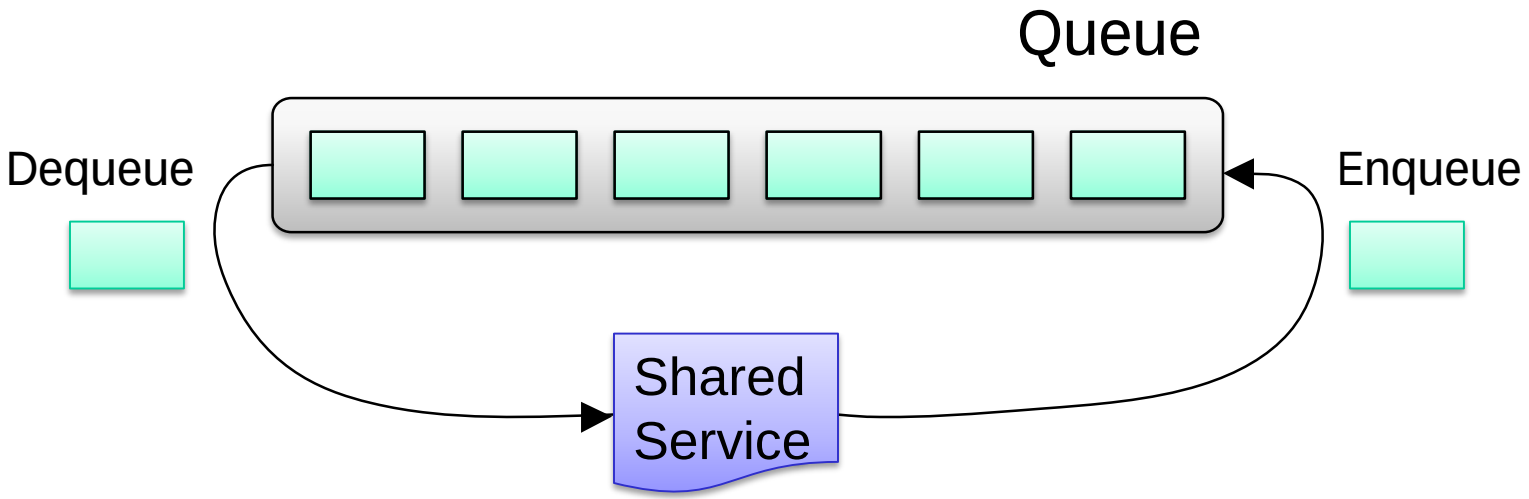
Slide by Matt Stallmann included with permission.



5.2절 큐

```
template <typename E>
class Queue {
public:
   int size() const;
   bool empty() const;
   const E& front() const throw(QueueEmpty);
   void enqueue(const E& e);
   void dequeue() throw(QueueEmpty);
};
```

활용예: Round Robin Scheduler



5.3절 데크

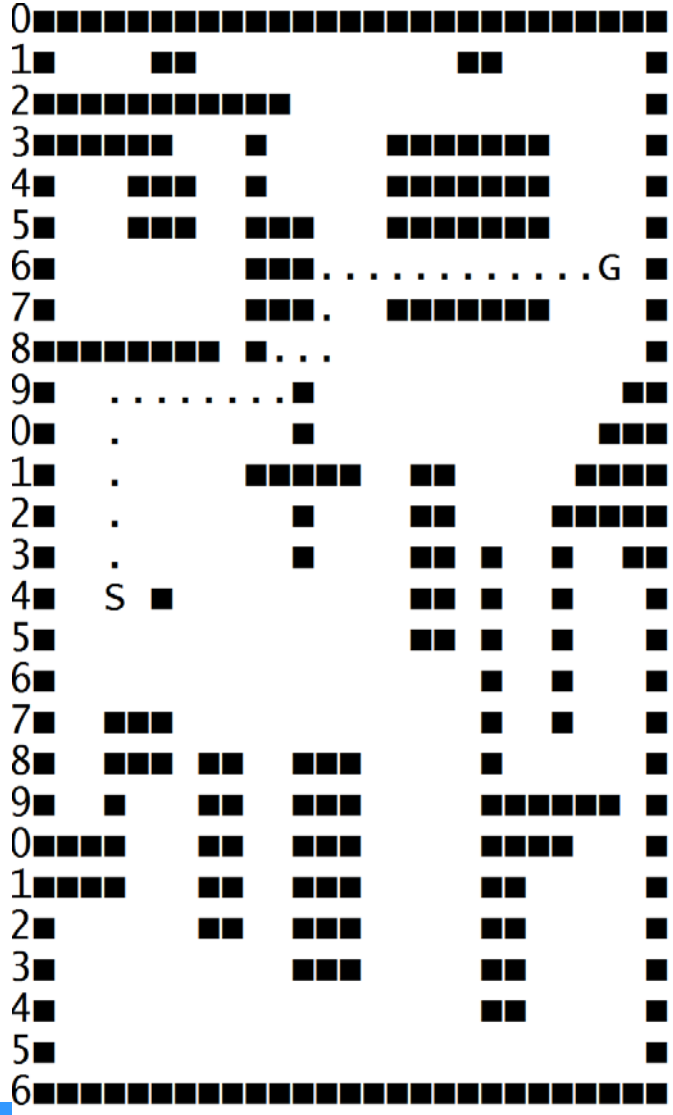
```
template <typename E>
class Deque {
public:
   int size() const;
   bool empty() const;
   const E& front() const  throw(DequeEmpty);
   const E& back() const  throw(DequeEmpty);
   void insertFront(const E& e);
   void insertBack(const E& e);
   void removeFront() throw(DequeEmpty);
   void removeBack()  throw(DequeEmpty);
};
```

Deque 활용

- Stack → Deque
  - top() → front()
  - push() → insertFront()
  - pop() → removeFront()
- Queue → Deque
  - front() → front()
  - enqueue() → insertBack()
  - dequeue() → removeFront()

활용예: Maze

$ ./Maze.exe q map1.txt



```
Complete path:
p(14,3) -> p(13,3) -> p(12,3) -> p(11,3) -> p(
10,3) -> p(9,3) -> p(9,4) -> p(9,5) -> p(9,6)
-> p(9,7) -> p(9,8) -> p(9,9) -> p(9,10) -> p(
8,10) -> p(8,11) -> p(8,12) -> p(7,12) -> p(6,
12) -> p(6,13) -> p(6,14) -> p(6,15) -> p(6,16
) -> p(6,17) -> p(6,18) -> p(6,19) -> p(6,20)
-> p(6,21) -> p(6,22) -> p(6,23) -> p(6,24)

Path Length (in points) : 30
Total Moves (in points) : 375
Maximum Reached Length of Open Queue: 22
Final Length of Open Queue (O): 9
Final Length of Closed Queue (C): 376
Total Number of Explored States (O+C): 385
```

Maze.cpp

```
// Taken from https://github.com/steffanc/MazeBot
// Edited by khkim for readability
#include <cstdlib>
#include <iostream>
#include <math.h>
#include <string>
#include <fstream>
//#define __STL_DEQUE__
#ifdef __STL_DEQUE__
#include <deque> // Standard Template Library
#else
#include "LinkedDeque.h" // "Data Structures and Algorithms in C++ (2nd Edition)"
#define deque LinkedDeque
#define push_back insertBack
#define pop_back removeBack
#define push_front insertFront
#define pop_front removeFront
#endif
```

Maze.cpp

```
class Point {
private:
  int depth;
  Point *parent;
public:
  int cy, cx;
  Point(): cy(0), cx(0), depth(0), parent(0) { }
  Point(int cy, int cx, int depth, Point * parent) { … }
  ~Point() { delete parent; }
  int getDepth() const { return depth; };
  Point *getParent() const { return parent; };
  friend ostream& operator<<(ostream& out, const Point& obj);
};
```

Maze.cpp

```
int main(int argc, char *argv[]) {
  Point start, goal;
  InitMaps(argv[1], start, goal);
  bool found = FindRoute(start, goal);
  if (!found) {
    cout << "This maze has no solution!" << endl;
    return 1;
  }

  PrintRoute();
  //ClearMaps();
  FreeMaps();
  return 0;
}
```

map1.txt

```
27 27
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 1
1 0 0 0 1 1 1 0 0 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 1
1 0 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 9 0 1
1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1
1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 0 0 1 1
1 0 0 8 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1
1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1
1 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1
1 0 0 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 1
1 1 1 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1
1 1 1 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1
1 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

- 27x27 행렬
- '1'→ 벽
- '0'→ 빈공간
- '8'→ 시작
- '9'→ 목표

감사합니다!