동국대학교
dongguk university

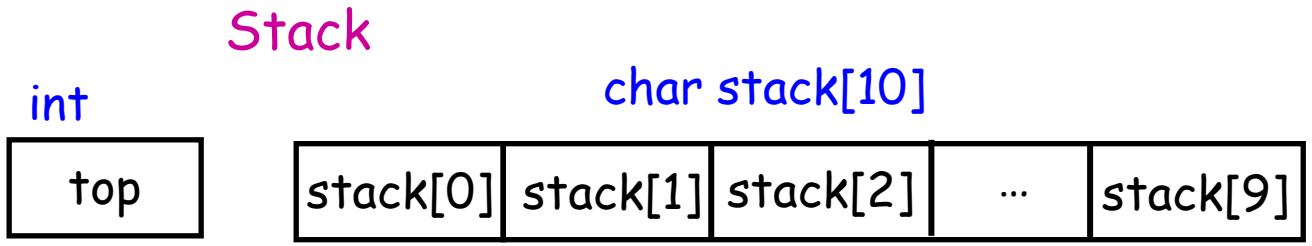
실습: Week 4

Data Structures

Contents

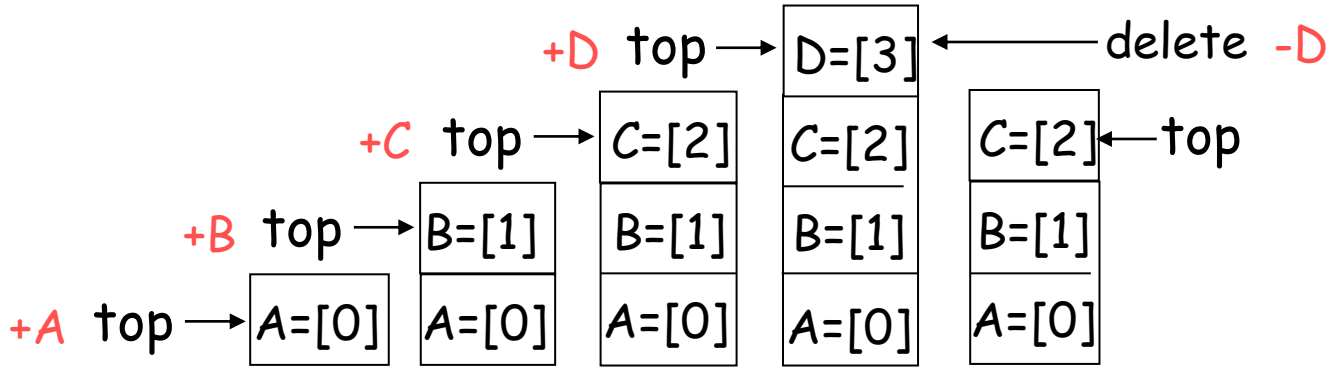
- Stack
  - Stack (array)
  - Postfix evaluation (use stack)
- 실습
  - 실습 4-1. Stack (array)구현
  - 실습 4-2. postfix evaluation program 구현

Stack 구조



top : 현재 위치 및 배열 크기 (초기값: -1)

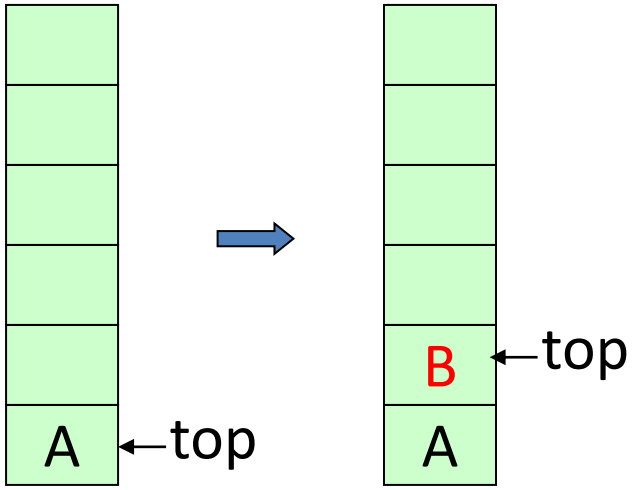
- LIFO (Last In First Out) 형식



Stack 연산

- 스택의 삽입 연산

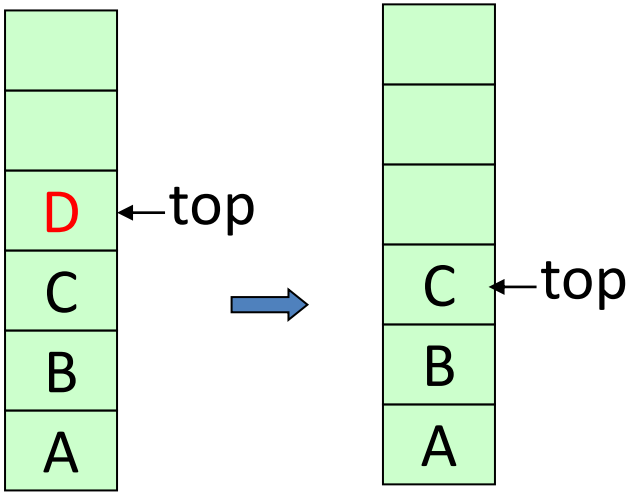
```
void push(Element  e)
{
        …
}
```



Stack 연산

- 스택의 삭제 연산

```
Element pop( )
{
        …
}
```



Postfix Evaluation

- 수식의 표현

  ((6/2)-3)+(4*2)

  - 중위 표기(*infix notation*): ((6/2)-3)+(4*2)
  - 후위 표기(*postfix notation*): 62/3-42*+
  - 전위 표기(*prefix notation*): +-/623*42

- 후위 표기법
  - 괄호 사용 안함
  - 연산자의 우선순위 없음 (L→R 순서의 계산)
    → 컴파일러가 사용

Postfix Evaluation

- 후위표기 연산방법
  - 연산자를 만날 때까지 피연산자를 스택에 저장
  - 연산자를 만나면 연산에 필요한 만큼의 피연산자를 스택에서 가져와서 연산을 실행하고 연산의 결과를 다시 스택에 저장
  - 예) 6 2 / 3 – 4 2 * +

토큰	stack[0]	stack[1]	stack[2]	top
6	6			0
2	6	2		1
/	6/2=3			0
3	3	3		1
-	3-3=0			0
4 0	4		1	
2	0	4	2	2
*	0	4*2=8		1
+	0+8=8			0

실습 4-1. Stack 구현

- 문자들의 스택을 테스트하는 프로그램 구현
- 명령어
  - push(+<문자>), pop(-)
  - 스택 내용을 출력(S)

실습 4-1. 실행예

```
******************** Command ********************
+<c>: push c, - : pop,
S: Show, Q: Quit
************************************************

Command> +
a
Command> +
b
Command> +
1
Command> +
2
Command> +
3
Command> s
a b 1 2 3
Command> -

 3

Command> -

 2

Command> -

 1

Command> -

 b

Command> -

 a

Command> -
stack is empty
```

자료구조 및 함수

- Element stack[MAX_SIZE];

  int top = -1;

- Void push(Element e)
  - Requires : 스택이 포화되지 않아야 함
  - Results : 스택에 e를 삽입
- Element pop()
  - Requires : 스택이 비어 있지 않아야 함
  - Results : 스택에서 원소를 반환
- Void stack_show()
  - Results : 스택의 내용을 보여줌

arraystack.h

```
#include <stdio.h>
#include <conio.h>

#define  MAX_SIZE 10
#define  boolean   unsigned char
#define  true      1
#define  false     0

typedef char Element;

// Global stack
Element stack[MAX_SIZE];
int top = -1;

void push(Element e);
Element pop();
void stack_show();
```

arraystack.c -main()함수

```
#include "arraystack.h"

void main() {

    char   c, e;


    printf("********************* Command ********************\n");
    printf("+<c>: push c, - : pop, \n");
    printf("S: Show, Q: Quit \n");
    printf("************************************************\n");
    while (1) {
            printf("\nCommand> ");
            c = _getche();
            c = toupper(c);
            printf("\n");
```

arraystack.c -main()함수

```
switch (c) {
            case '+':
                        e = _getche();
                        push(e);
                        break;
            case '-':
                        e = pop();
                        if (e != NULL)
                                    printf("\n %c \n", e);
                        break;
            case 'S':
                        stack_show();
                        break;
            case 'Q':
                        printf("\n");
                        exit(1);
            default: break;
            }
    }
}
```

실습 4-2. postfix evaluation

- 후위 연산식 계산 함수 구현
  - 후위연산식 하나를 문자열로 받아 계산 결과를 반환
  - 스택 사용
  - 연산자 : +, -, *, /
  - 나누기(/) 주의 사항
    - 0으로 나누는 경우 예외처리
    - int형으로 반환
  - 연산자 부족, 숫자 및 연산자 외 다른 데이터 입력 등 문제는 고려하지 않음.

실습 4-2. 실행 예

```
Input postfix expression: 123*+
Result = 7

Input postfix expression: 62/3-42*+
Result = 8

Input postfix expression: 83+
Result = 11

Input postfix expression: 83-
Result = 5

Input postfix expression: 83/
Result = 2

Input postfix expression: 83*
Result = 24

Input postfix expression: 60/
divided by 0 error
```

자료구조 및 함수

- int stack[MAX_SIZE];
  int top = -1;
  boolean err_check = false; # error 발생 확인

- int eval_postfix(char *exp)
  - Results : exp(후위표기 연산식)를 처리하여 결과를 반환
- void push(int e)
- int pop( )
- boolean is_number(char c);
- boolean is_op(char c);

eval_postfix.h

```
#include <stdio.h>
#pragma warning(disable : 4996)


#define  MAX_SIZE 100
#define  boolean   unsigned char
#define  true      1
#define  false     0

// Global stack
int stack[MAX_SIZE];
int top = -1;
boolean err_check = false;

void push(int e);
int pop();

int eval_postfix(char* exp);
boolean is_number(char c);
boolean is_op(char c);
```

eval_postfix.c - main()함수

```
#include "eval_postfix.h"

void main()
{
        char exp[100];  // postfix expression
        int result;

        while (1) {
                printf("\n Input postfix expression: ");
                scanf("%s", exp);

                result = eval_postfix(exp);
                if (err_check == false)
                        printf(" Result = %d \n\n", result);
        }
}
```

eval_postfix.c - 그 외 함수

```
boolean is_number(char c)
{
    if (('0' <= c) && (c <= '9'))
        return true;
    else
        return false;
}
```