

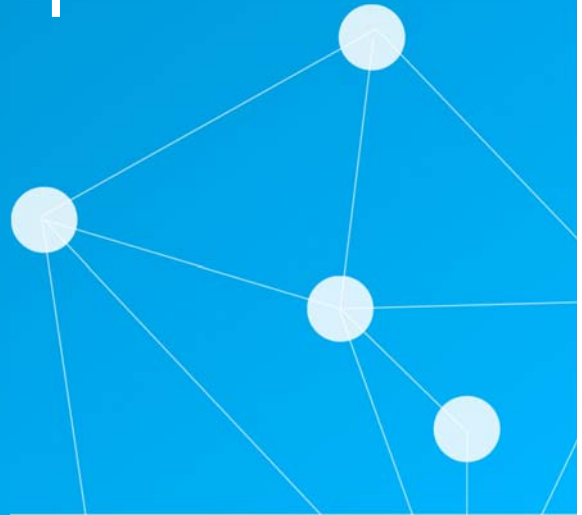
- 스택의 개념과 추상 자료형을 이해한다.
- 스택의 동작 원리를 이해한다.
- 배열을 이용한 스택의 구현 방법을 이해한다.
- 괄호 검사, 수식의 계산, 미로 탐색 등에 스택을 활용하여 문제를 해결할 수 있는 능력을 배양한다.

03

CHAPTER

스택

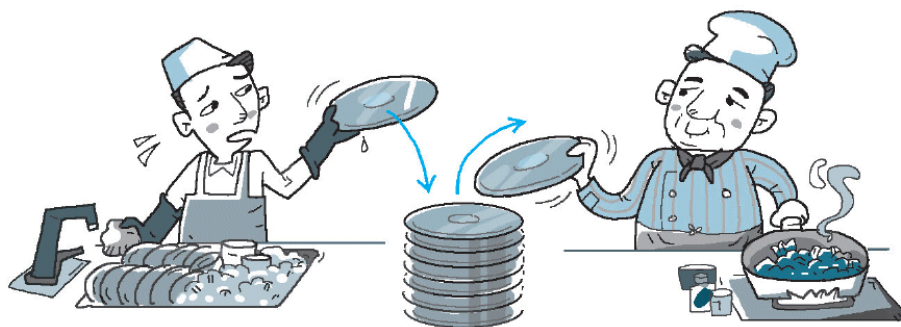
- 3.1 스택이란?
- 3.2 배열을 이용한 스택
- 3.3 스택의 구현: int 스택
- 3.4 스택의 구현: 구조체를 저장하는 스택
- 3.5 괄호 검사
- 3.6 후위 표기 수식의 계산
- 3.7 중위 표기 수식의 후위 표기 변환



스택이란?



- 스택(stack): 쌓아놓은 더미
- 후입선출(LIFO: Last-In First-Out)
 - 가장 최근에 들어온 데이터가 가장 먼저 나감





- 10, 20, 30, 40, 50을 스택에 넣었다가 3개의 항목을 삭제하였다. 남아 있는 항목은?

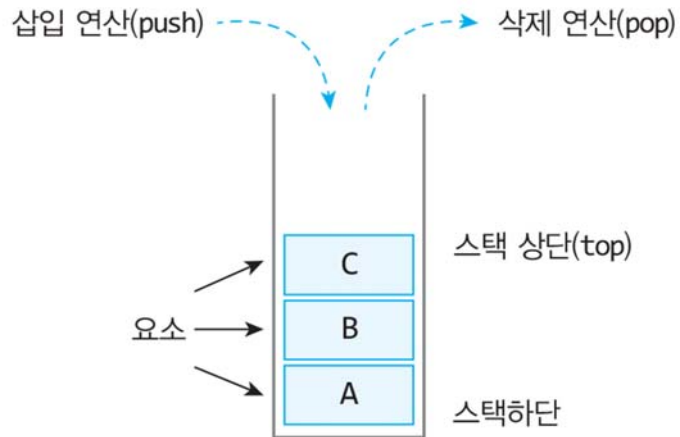


- 순서가 A, B, C, D로 정해진 입력 자료를 스택에 입력하였다가 출력할 때, 가능한 출력 순서의 결과가 아닌것은?
 1. D, A, B, C
 2. A, B, C, D
 3. A, B, D, C
 4. B, C, D, A

스택의 구조



- 스택 상단: **top**
- 스택 하단: 불필요
- 요소(element), 항목
- 공백상태, 포화상태
- **삽입, 삭제 연산**



5

스택의 추상 자료형



- 객체: 무엇이든 가능 (닭은 접시들)
- 연산 (접시함의 경우)
 - 닭은 접시를 보관함에 넣음 -> 새로운 항목을 스택에 삽입
 - 보관함에서 접시 하나를 꺼냄 -> 하나의 항목을 꺼냄.
 - 보관함에 접시가 있는지 살핌 -> 스택이 비어있는지 살핌
- 고급 접시함 ➔ 고급 기능 추가
 - 보관함 내에 어떤 접시가 있는지 모니터에 출력해준다.
 - 보관함이 가득 차 있는지를 살핀다.
 - 보관함 내의 접시 개수를 알려준다.
 - 접시를 꺼내지 않고 맨 위의 접시가 무엇인지 알아본다.

6

스택 추상 자료형



- Stack ADT

데이터: 후입선출(LIFO)의 접근 방법을 유지하는 요소들의 모음

연산:

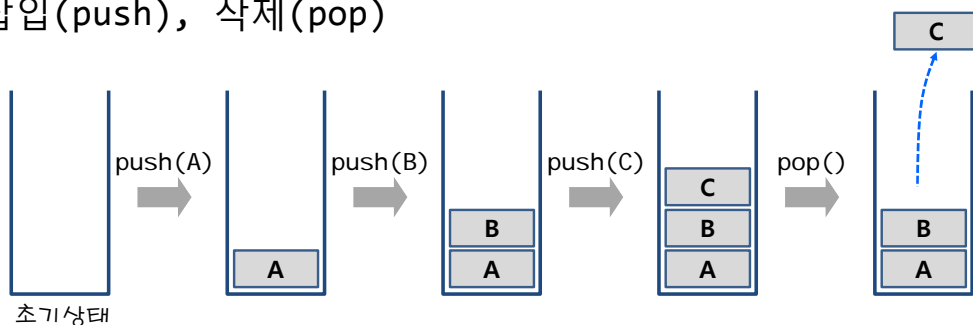
- `init()`: 스택을 초기화한다.
- `is_empty()`: 스택이 비어있으면 TRUE를 아니면 FALSE를 반환한다.
- `is_full()`: 스택이 가득 차 있으면 TRUE를 아니면 FALSE를 반환한다.
- `size()`: 스택내의 모든 요소들의 개수를 반환한다.
- `push(x)`: 주어진 요소 `x`를 스택의 맨 위에 추가한다.
- `pop()`: 스택 맨 위에 있는 요소를 삭제하고 반환한다.
- `peek()`: 스택 맨 위에 있는 요소를 삭제하지 않고 반환한다.

7

스택의 연산



– 삽입(push), 삭제(pop)

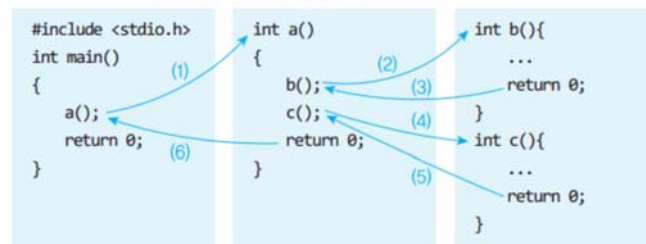


- `is_empty()`: 스택이 공백상태인지 검사
- `is_full()`: 스택이 포화상태인지 검사
- `peek(s)`: 요소를 스택에서 삭제하지 않고 보기만 하는 연산
 - (참고) `pop` 연산은 요소를 스택에서 완전히 삭제하면서 가져옴.

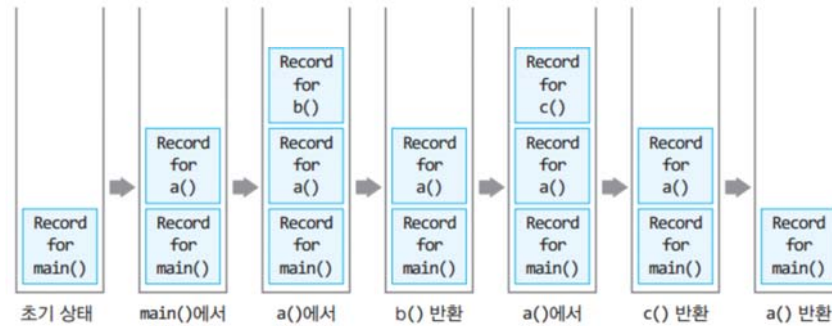
8

스택의 용도

- 함수호출



- Undo기능
- 괄호검사
- 계산기
- 미로탐색 등

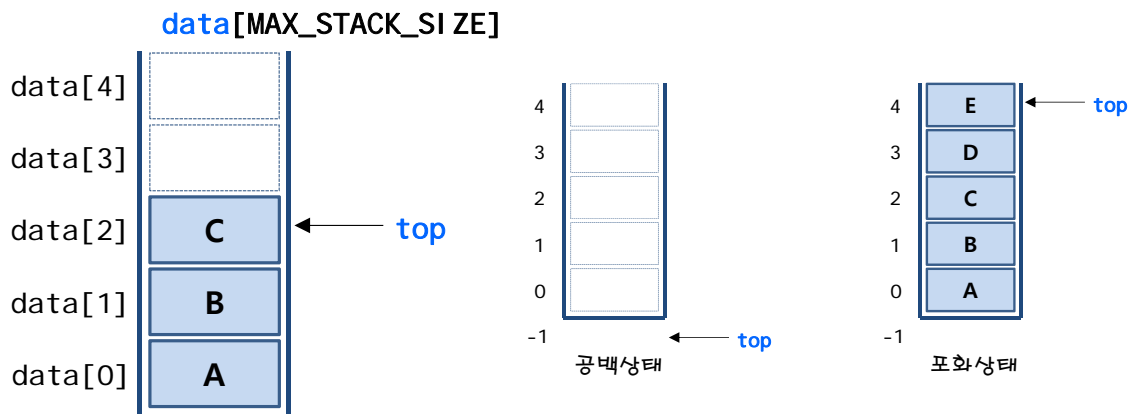


9

- 스택과 같은 입출력 형태를 _____(LIFO) 라고 한다.
- 스택을 사용하여 a, b, c, d, e를 입력하고 b, c, e, d, a와 같은 출력을 얻으려면 입력과 출력의 순서를 어떻게 해야 하는가?

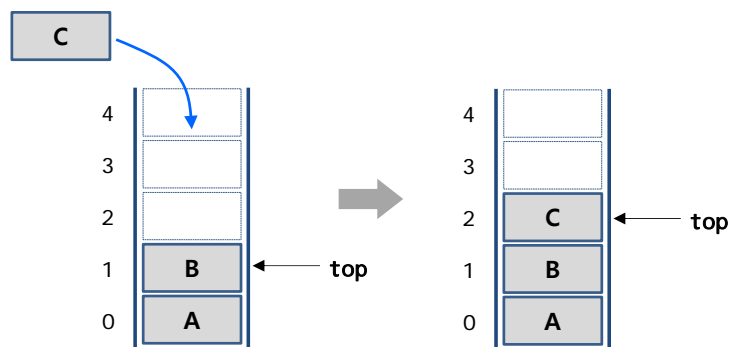
3.2 배열을 이용한 스택의 구현

- 1차원 배열 `stack[]`
 - `top`: 가장 최근에 입력되었던 자료를 가리키는 변수
 - `stack[0]... stack[top]`: 먼저 들어온 순으로 저장
 - 공백상태이면 `top`은 -1



11

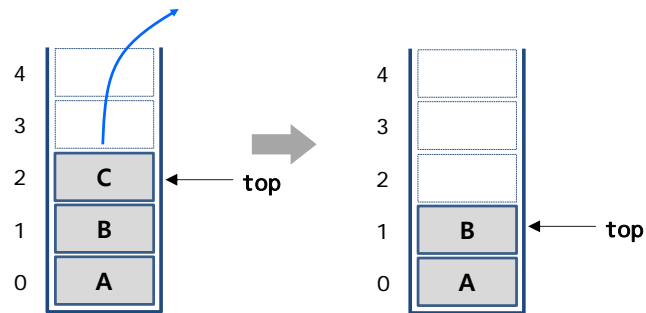
Push 연산



```
push(x)
if is_full(S)
    then error "overflow"
    else top ← top + 1
        data[top] ← x
```

12

Pop 연산



```
pop()
if is_empty()
  then error "underflow"
else e ← data[top]
     top ← top-1
     return e
```

13

- 스택에서 삽입 작업이 발생하면 top의 값은?
 1. $top == 0$
 2. $top == 1$
 3. $top = top - 1$
 4. $top = top + 1$

- 다음 중 스택에 대한 설명 중 맞는 것은?
 1. 스택은 FIFO(First In First Out) 방식으로 동작한다.
 2. 스택은 양쪽 끝을 사용하여 입출력을 한다.
 3. 스택의 삭제 연산보다 스택의 삽입 연산이 훨씬 쉽다.
 4. 스택은 중간에서 요소를 삭제하는 것을 허용하지 않는다.



- 스택에서 top이 가리키는 것은 무엇인가?
- top이 8이면 현재 스택에 저장된 데이터의 총수는 몇 개인가?
- 다음 중 배열로 구현된 스택에서 공백 상태에 해당하는 조건은? 또 포화 상태에 해당하는 조건은?
 1. $top == -1$
 2. $top == 0$
 3. $top == (MAX_STACK_SIZE - 1)$
 4. $top == MAX_STACK_SIZE$



- 스택에 항목들을 삽입하고 삭제하는 연산은 시간 복잡도가 어떻게 되는가?
 1. $O(1)$
 2. $O(\log_2 n)$
 3. $O(n)$
 4. $O(n^2)$



- 다음과 같이 스택을 사용하는 프로그램이 하는 일은 무엇인가?

```
init_stack();
int n = 4096;
while (n > 0) {
    push(n % 2);
    n /= 2;
}
while (!is_empty())
    printf("%d", pop());
```

3.3 스택의 구현



- 데이터
 - data[], top → 전역 변수로 선언

```
typedef int Element;  
Element data[MAX_STACK_SIZE];  
int top;
```

```
int is_empty()  
{  
    if( top == -1 )  
        return 1;  
    else  
        return 0;  
}
```

- 간단한 함수

```
void init_stack() { top = -1; }  
int size() { return top+1; }  
int is_empty() { return (top == -1); }  
int is_full() { return (top == MAX_STACK_SIZE-1); }
```

19

스택의 주요 함수



```
void push ( Element e )  
{  
    if( is_full() )  
        error ("스택 포화 에러");  
    data[++top] = e;  
}
```

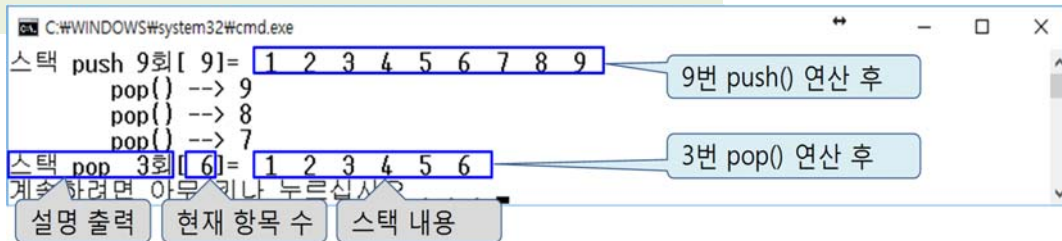
```
Element pop ( )  
{  
    if( is_empty() )  
        error ("스택 공백 에러");  
    return data[top--];  
}  
Element peek ( )  
{  
    if( is_empty() )  
        error ("스택 공백 에러");  
    return data[top];  
}
```

```
void print_stack(char msg[]) {  
    int i;  
    printf("%s[%2d]= ", msg, size());  
    for (i=0 ; i<size() ; i++)  
        printf("%2d ", data[i]);  
    printf("\n");  
}
```

20

사용방법

```
void main()
{
    int i;
    init_stack();
    for( i=1 ; i<10 ; i++ )
        push( i );
    print_stack("스택 push 9회");
    printf("\tpop() --> %d\n", pop());
    printf("\tpop() --> %d\n", pop());
    printf("\tpop() --> %d\n", pop());
    print_stack("스택 pop 3회");
}
```



21

- 사용자로부터 문자열을 읽고 이것을 역순으로 출력하는 프로그램 reverse.c를 작성하고 실행 예를 보여라. 스택을 사용한다.

22



- 배열을 이용한 스택의 문제점은 스택을 생성할 때 MAX_STACK_SIZE를 결정해야 한다는 점이다. 이 결점을 보완하는 한 가지 방법은 max_top이라는 변수를 도입하여 스택이 만들어질 때는 max_top을 0으로 하여 시작하고, 만약 삽입 연산 중에 새로운 요소를 추가할 공간이 없을 때에는 max_top을 max_top*2+1로 변경하고 변경된 크기만큼의 공간을 동적으로 할당하여 새로운 배열을 만든다. 요소들은 이전 배열에서 새로운 배열로 복사되고 이전 배열은 지워진다. 비슷하게 만약 삭제 연산 중에 요소들의 개수가 배열 크기의 1/4로 떨어지게 되면 기존 크기의 절반인 배열을 생성하고 이전 요소들을 복사한 다음, 이전의 배열을 삭제한다. 이 스택을 구현하고 테스트한 예를 보여라.

```
typedef struct {
    element *stack;
    int top;
    int max_top;    // 현재 배열의 크기
} RStackType;
```

3.4 구조체를 저장하는 스택



- 학생정보스택
 - 구조체 정의
 - Element 정의

```
typedef struct Student_t {
    int    id;
    char    name[32];
    char    dept[32];
} Student;

typedef Student Element;
```

- 출력 함수 수정

```
void print_stack(char msg[])
{
    int i;
    printf("%s[%2d]= ", msg, size()) ;
    for (i=0 ; i<size() ; i++ )
        printf("\n\t%-15d %-10s %-20s",
                data[i].id, data[i].name, data[i].dept);
    printf("\n");
}
```

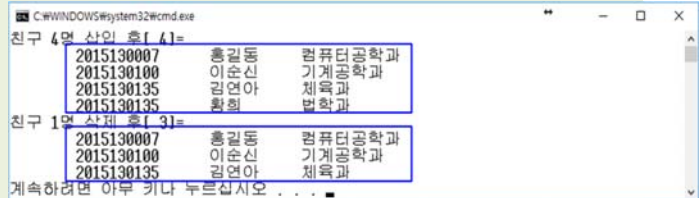
학생정보스택 프로그램

```
Student get_student(int id, char* name, char* dept)
```

```
{  
    Student s;  
    s.id = id;  
    strcpy(s.name, name);  
    strcpy(s.dept, dept);  
    return s;  
}
```

```
void main()  
{
```

```
    init_stack( );  
    push( get_student(2015130007, "홍길동", "컴퓨터공학과") );  
    push( get_student(2015130100, "이순신", "기계공학과") );  
    push( get_student(2015130135, "김연아", "체육과") );  
    push( get_student(2015130135, "황희", "법학과") );  
    print_stack("친구 4명 삽입 후");  
    pop( );  
    print_stack("친구 1명 삭제 후");  
}
```



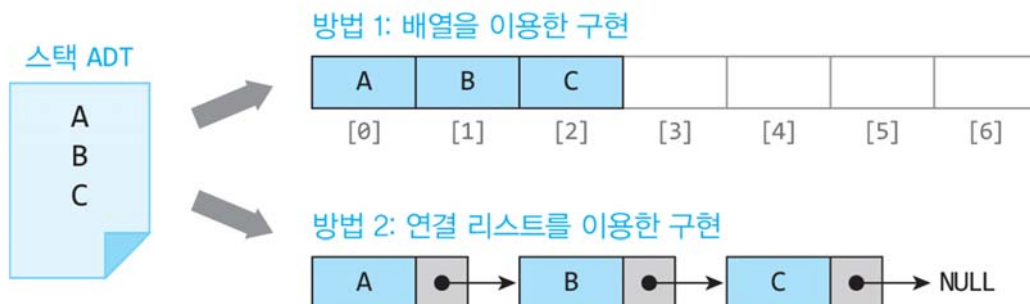
2015130007	홍길동	컴퓨터공학과
2015130100	이순신	기계공학과
2015130135	김연아	체육과
2015130135	황희	법학과

2015130007	홍길동	컴퓨터공학과
2015130100	이순신	기계공학과
2015130135	김연아	체육과

25

스택 구현의 다른 방법

- 연결 리스트를 이용하는 방법



26

3.5 스택 응용 : 괄호 검사



- 괄호의 종류: 대중소 ('[', ']'), ('{', '}'), ('(', ')')
- 조건
 1. 왼쪽 괄호의 개수와 오른쪽 괄호의 개수가 같아야 한다.
 2. 왼쪽 괄호는 오른쪽 괄호보다 먼저 나와야 한다.
 3. 괄호 사이에는 포함 관계만 존재한다. 교차하면 안된다.
- 잘못된 괄호 사용의 예
 - (a(b)
 - a(b)c)
 - a{b(c[d]e)f)

27

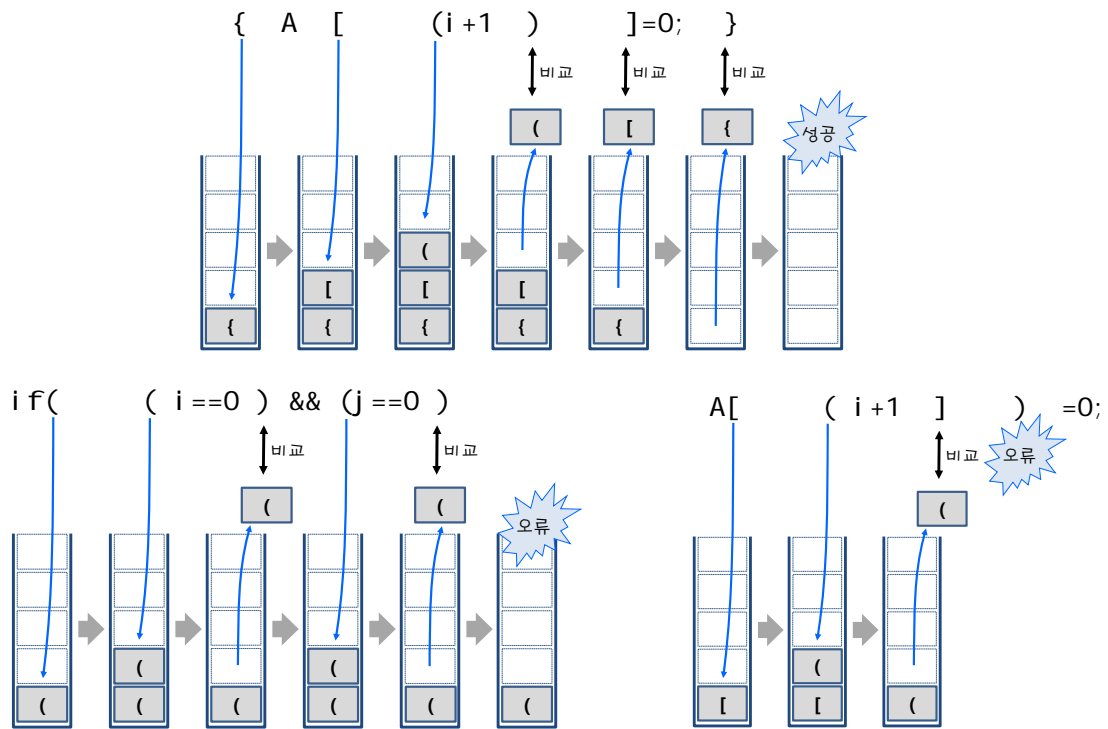
괄호검사 알고리즘



- 알고리즘의 개요
 - 문자열에 있는 괄호를 차례대로 조사하면서 왼쪽 괄호를 만나면 스택에 삽입하고, 오른쪽 괄호를 만나면 스택에서 top 괄호를 삭제한 후 오른쪽 괄호와 짝이 맞는지를 검사한다.
 - 이 때, 스택이 비어 있으면 조건 1 또는 조건 2 등을 위배하게 되고 괄호의 짝이 맞지 않으면 조건 3 등에 위배된다.
 - 마지막 괄호까지를 조사한 후에도 스택에 괄호가 남아 있으면 조건 1에 위배되므로 0(거짓)을 반환하고, 그렇지 않으면 1(참)을 반환한다.

28

괄호 검사 예



29

- 다음과 같은 수식이 주어졌을 경우, 알고리즘을 추적해서 각 단계에서의 스택의 내용을 그려라.

`a { b [(c + d) * e] - f }`

30

괄호검사 알고리즘

`check_matching(expr)`

```
while (입력 expr의 끝이 아니면)
  ch ← expr의 다음 글자
  switch(ch)
    case '(': case '[': case '{':
      ch를 스택에 삽입
      break
    case ')': case ']': case '}':
      if (스택이 비어 있으면)
        then 오류
      else 스택에서 open_ch를 꺼낸다
        if (ch와 open_ch가 같은 짝이 아니면)
          then 오류 보고
        break
  if(스택이 비어 있지 않으면)
    then 오류
```

왼쪽 괄호이면
스택에 삽입

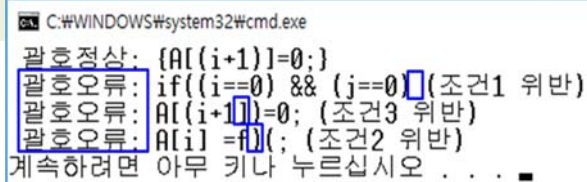
오른쪽 괄호이면
스택에서 삭제비교

31

괄호검사 테스트 프로그램

```
void main()
{
  char expr[4][80] = { "{A[(i+1)]=0;}", "if((i==0) && (j==0)",
    "A[(i+1)]=0;", "A[i] =f)(;" };

  int err, i;
  for( i=0 ; i<4 ; i++ ) {
    err = check_matching(expr[i]);
    if( err == 0 )
      printf(" 괄호정상: %s\n", expr[i]);
    else
      printf(" 괄호오류: %s (조건%d 위반)\n", expr[i], err);
  }
}
```



```
C:\WINDOWS\system32\cmd.exe
괄호정상: {A[(i+1)]=0;}
괄호오류: if((i==0) && (j==0) (조건1 위반)
괄호오류: A[(i+1)]=0; (조건3 위반)
괄호오류: A[i] =f)(; (조건2 위반)
계속하려면 아무 키나 누르십시오 . . .
```

32



- 회문(palindrome)이란 앞뒤 어느 쪽에서 읽어도 똑같은 단어나 문장을 의미한다. 예를 들면 "eye", "madam, I'm Adam", "race car" 등이다. 여기서 물론 구두점이나 스페이스, 대소문자 등은 무시하여야 한다. 스택을 이용하여 주어진 문자열이 회문인지 아닌지를 결정하는 프로그램을 작성하고 실행 예를 보이시오.

3.6 수식의 계산



- 수식의 표기방법:
 - 전위(prefix), 중위(infix), 후위(postfix)

중위 표기법	전위 표기법	후위 표기법
$2+3*4$	$+2*34$	$234*+$
$a*b+5$	$+5*ab$	$ab*5+$
$(1+2)+7$	$+7+12$	$12+7+$

- 컴퓨터에서의 수식 계산순서
 - 중위표기식-> 후위표기식->계산
 - $2+3*4 \rightarrow 234*+ \rightarrow 14$
 - 모두 스택을 사용

후위 표기 수식 계산

알고리즘

Calc_postfix (expr)

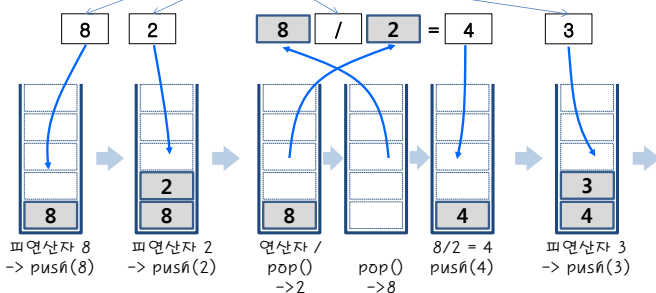
```

스택 초기화;
for 항목 in expr
do if (항목이 피연산자이면)
    s.push(item);
    if (항목이 연산자 op이면)
    then second ← pop();
        first ← pop();
        temp ← first op second; // op 는 +-* / 중의 하나
        push(temp);
result ← pop();
    
```

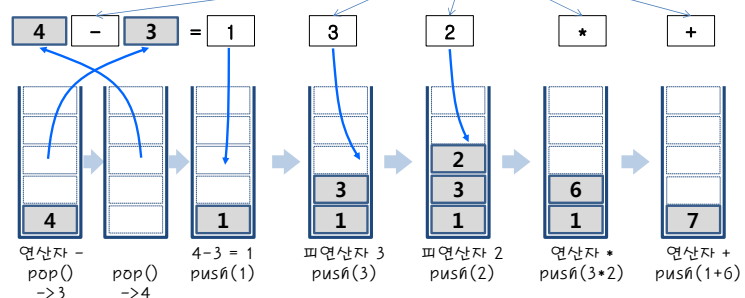
35

후위 표기 수식 계산 예

후위 표기 수식: 8 2 / 3 - 3 2 * +



후위 표기 수식: 8 2 / 3 - 3 2 * +



36

후위 표기 수식 계산 프로그램

```
double calc_postfix( char expr[] ) {
    ...
    init_stack();
    while( expr[i] != '\0' ) {
        c = expr[i++];
        if (c>='0' && c<='9') {
            val = c - '0';
            push( val );
        }
        else if( c=='+' || c=='-' || c=='*' || c=='/' ){
            ...
            switch( c ) {
                case '+': push( val1 + val2); break;
                ...
            }
        }
    }
}

void main() {
    char expr[2][80] = { "8 2 / 3- 3 2 * +", "1 2 / 4 * 1 4 / *" };
    printf("수식: %s = %f\n", expr[0], calc_postfix(expr[0]));
    printf("수식: %s = %f\n", expr[1], calc_postfix(expr[1]));
}
```



C:\WINDOWS\system32\cmd.exe

수식: 8 2 / 3- 3 2 * + = 7.000000
수식: 1 2 / 4 * 1 4 / * = 0.500000
계산 결과
계속하려면 아무 키나 누르십시오 . . .

37

3.7 중위 표기 수식의 후위 표기 변환

- 중위표기와 후위표기
 - 중위와 후위 표기법의 공통점: 피연산자의 순서가 동일
 - 연산자들의 순서만 다름(우선순위순서, 괄호)
 - 연산자만 스택에 저장했다가 출력
 - $2+3*4 \rightarrow 234*+$
- 알고리즘
 - 피연산자를 만나면 그대로 출력
 - 연산자를 만나면 **우선순위가 같거나 높은 스택의 연산자를 출력** 아니면 스택에 저장
 - 왼쪽 괄호는 우선순위가 가장 낮은 연산자로 취급
 - 오른쪽 괄호가 나오면 스택에서 왼쪽 괄호 위의 쌓여있는 모든 연산자를 출력하고 왼쪽 괄호 삭제
 - 입력 수식이 끝나면 모두 pop()

38

중위 → 후위 표기 변환 예



중위 표기 수식	연산자 스택	후위 표기 수식	중위 표기 수식	연산자 스택	후위 표기 수식
A + B * C			A * B + C		
A + B * C		A	A * B + C		A
A + B * C	+	A B	A * B + C	*	A B
A + B * C	+	A B C	A * B + C	*	A B C
A + B * C 연산자 우선순위 비교 (* > +) *를 바로 삽입	*	A B C	A * B + C 연산자 우선순위 비교 (+ <= *) 우선순위가 같거나 높은 +를 먼저 출력 후 + 삽입	+	A B C
A + B * C	+	A B C	A * B + C	+	A B C
A + B * C		A B C * +	A * B + C		A B C * +

39

후위 표기 변환 예



중위 표기 수식	연산자 스택	후위 표기 수식
(A + B) * C		
(A + B) * C 괄호는 일단 삽입	(
(A + B) * C	(A
(A + B) * C	(A
(A + B) * C 괄호는 우선순위가 가장 낮음 -> + 삽입	+	A B
(A + B) * C	+	A B
(A + B) * C)'가 나오면 '(' 전까지 모두 출력 괄호는 후위 표기식에 출력하지 않음)	A B +
(A + B) * C	*	A B +
(A + B) * C	*	A B + C
(A + B) * C		A B + C *

40

후위 표기 변환 알고리즘



Infix_to_postfix(expr)

스택 초기화.

while (expr에 처리할 항이 남아 있으면)

term ← 다음에 처리할 항;

switch (term)

case 피연산자:

term을 출력;

break;

case 왼쪽 괄호:

push(term);

break;

case 오른쪽 괄호:

e ← pop();

while(e ≠ 왼쪽 괄호)

do e를 출력;

e ← pop();

break;

case 연산자:

while (peek()의 우선순위 ≥ term의 우선순위)

do e ← pop();

e를 출력;

push(term);

break;

while(not is_empty())

do e ← pop();

e를 출력;

41

후위 표기 변환 프로그램



```
static int precedence( char op ) {
    switch (op) {
        case '(' : case ')' : return 0;
        case '+' : case '-' : return 1;
        case '*' : case '/' : return 2;
    }
    return -1;
}

void infix_to_postfix( char expr[] ) {...}
void main()
{
    char expr[2][80] = { "8/2-3+(3*2)", "1/2* 4 * (1/4)" };
    printf("중위수식: %s ==> 후위수식:", expr[0]);
    infix_to_postfix(expr[0]);
    printf("중위수식: %s ==> 후위수식:", expr[1]);
    infix_to_postfix(expr[1]);
}
```

C:\WINDOWS\system32\cmd.exe

중위수식: 8 / 2 - 3 + (3 * 2) ==> 후위수식: 8 2 / 3 - 3 2 * +
중위수식: 1 / 2 * 4 * (1 / 4) ==> 후위수식: 1 2 / 4 * 1 4 / *
계속하려면 아무 키나 누르십시오 . . .

후위 표기 변환 결과

42

실습



- 중위 수식을 입력받아 후위 표기로 변환한 후 이를 다시 후위 표기 수식 계산 방법으로 계산하여 결과를 출력하는 프로그램을 작성하라.
 - 프로그램 3.4와 3.5를 결합