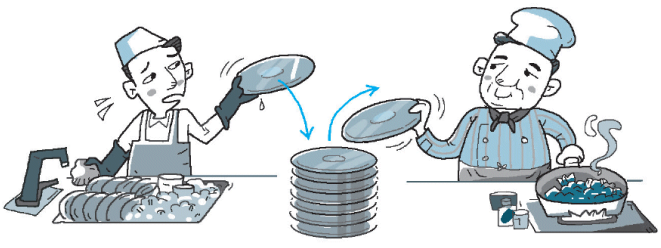
DATA
STRUCTURES
USING C

03 CHAPTER

스택

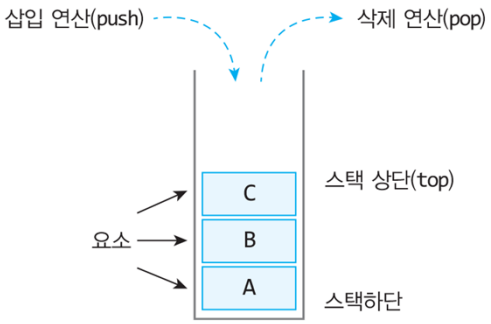
스택이란?

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



스택의 구조

- 스택 상단: top
- 스택 하단: 불필요
- 요소, 항목
- 공백상태, 포화상태
- 삽입, 삭제연산



3

스택 추상 자료형

- Stack ADT

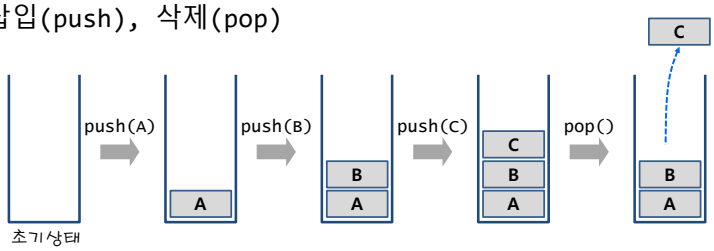
데이터: 후입선출(LIFO)의 접근 방법을 유지하는 요소들의 모음

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

4

스택의 연산

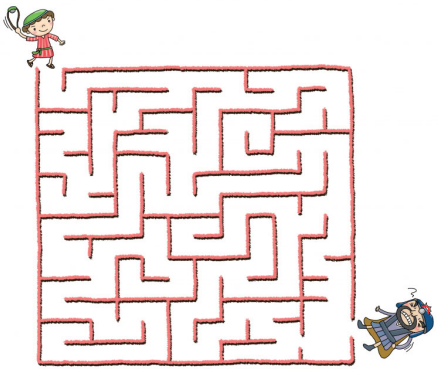
- 삽입(push), 삭제(pop)



- is_empty(): 스택이 공백상태인지 검사
- is_full(): 스택이 포화상태인지 검사
- peek(s): 요소를 스택에서 삭제하지 않고 보기만 하는 연산
  - (참고) pop 연산은 요소를 스택에서 가져옴, Top의 위치이동

스택의 활용

- 함수호출
- Undo기능
- 괄호검사
- 계산기
- 미로탐색 등

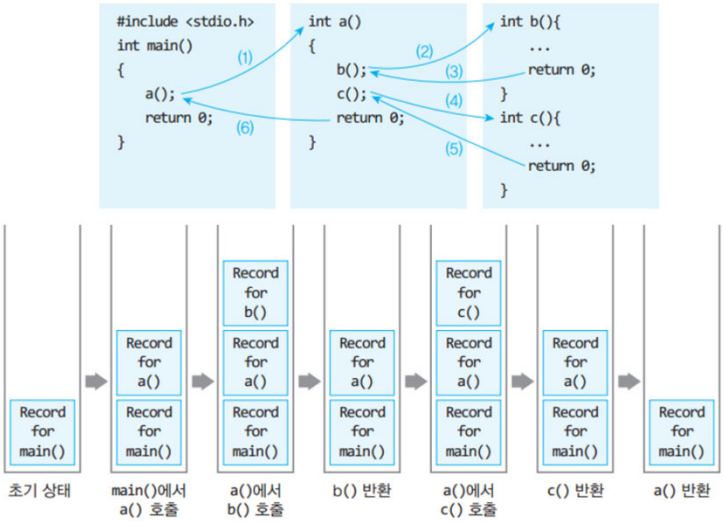


< 미로찾기 >

http://picok.co.kr/picok/image/view.php?it_id=m13802535752979

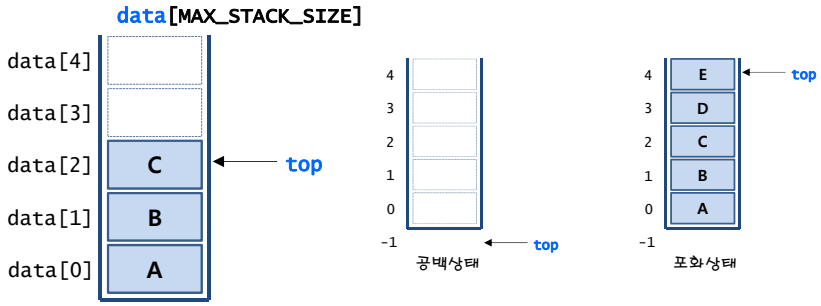
스택의 활용

– 함수호출

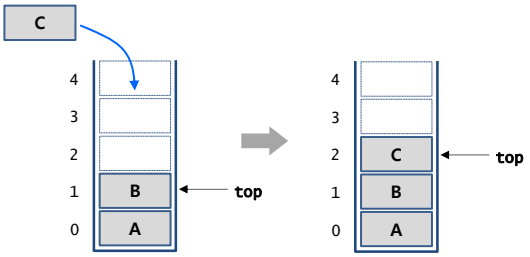


배열을 이용한 스택의 구현

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



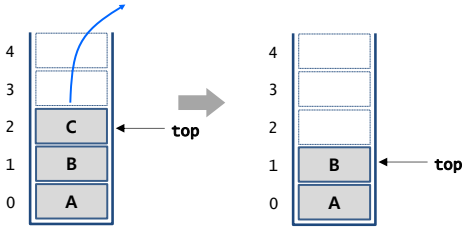
Push 연산




```
push(x)

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

9

Pop 연산




```
pop()

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

10

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); }
```

11

스택의 주요 함수

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

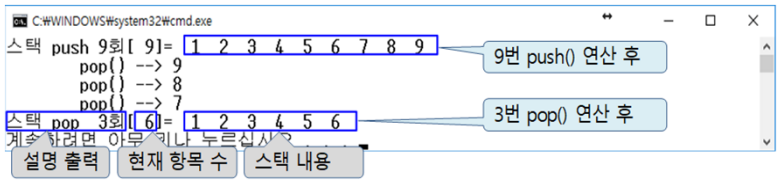
```
Element pop ( )
{
    if( is_empty() )
        error ("스택 공백 에러");
    return data[top--];
}
Element peek ( )
{
    if( is_empty() )
        error ("스택 공백 에러");
    return data[top];
}
```

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

12

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");
}
```



구조체를 저장하는 스택

- 학생정보스택
  - 구조체 정의
  - Element 정의

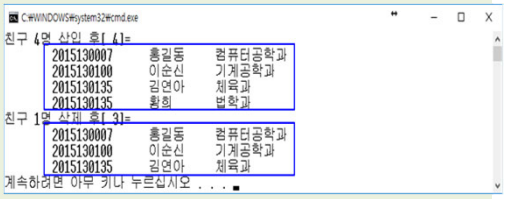
```
typedef struct Student_t {
        int     id;
        char    name[32];
        char    dept[32];
} Student;

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");
}
```

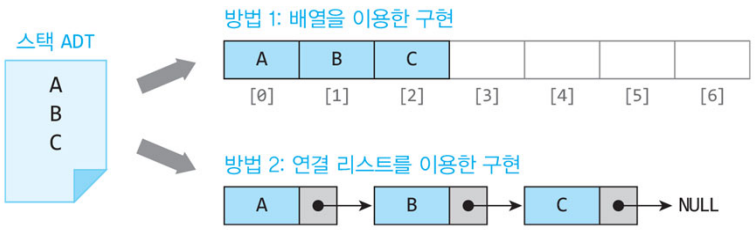
학생정보스택 프로그램

```
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");
}
```



스택 구현의 다른 방법

- 연결 리스트를 이용하는 방법

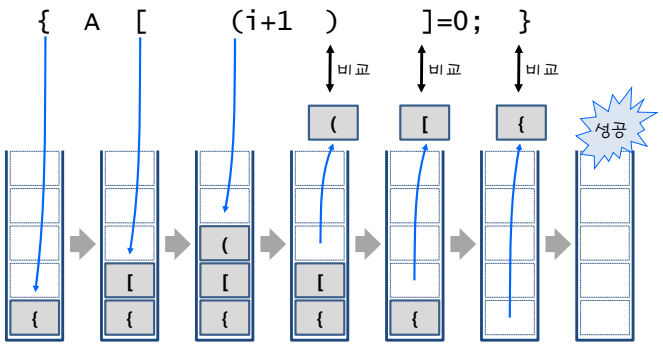


스택 응용 : 괄호 검사

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

17

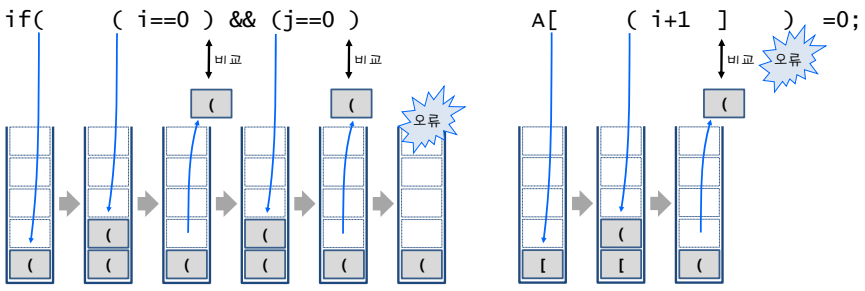
괄호 검사 예



< 바른 예 >

18

괄호 검사 예



< 잘못 사용 예 >

괄호검사 알고리즘

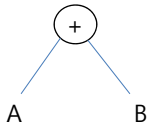
```
check_matching(expr)

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 (ch 와 open_ch가 같은 짝이 아니면)
                 then 오류 보고
       break
if( 스택이 비어 있지 않으면 )
  then 오류
```

왼쪽 괄호이면 스택에 삽입

오른쪽 괄호이면 스택에서 삭제비교

수식의 표현

► A + B 의 경우



1. Prefix : Preorder에 따른 순회(R – 좌 – 우)
   예) + A B

2. Infix : Inorder에 따른 순회(좌 – R – 우)
   예) A + B

3. Postfix : Postorder에 따른 순회(좌 – 우 - R)
   예) A B +

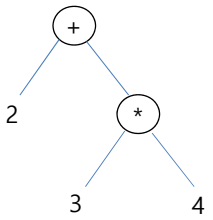
21

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 -> 234*+ -> 14
  - 모두 스택을 사용



22

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();
```

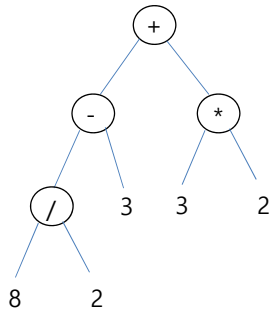

스택의 활용

► (8 / 2 – 3) + (3 * 2) 의 경우

1) Infix
   : 8 / 2 – 3 + 3 * 2

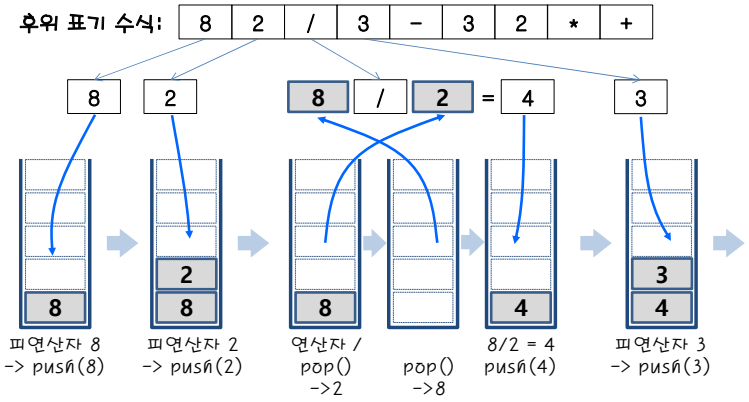
2) Prefix
   : + - / 8 2 3 * 3 2

3) Postfix
   : 8 2 / 3 – 3 2 * +

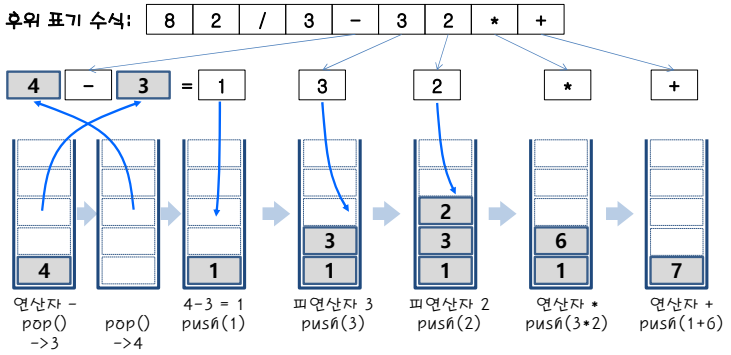


※ 컴파일러에서 중위표기를 후위표기로 변환 후, 스택을 사용하여 연산수행
후위표기로 사용시 연산자의 우선순위 고려와 괄호 없이도 연산이 가능함

후위 표기 수식 계산 예



후위 표기 수식 계산 예(계속)

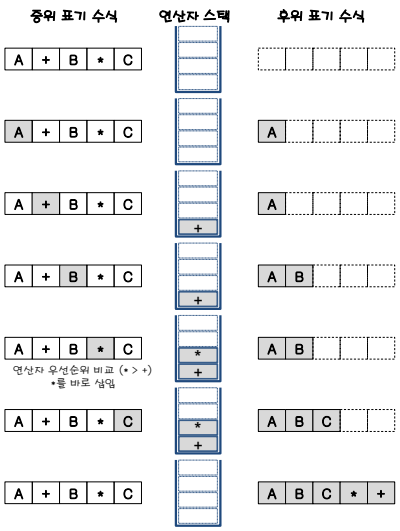


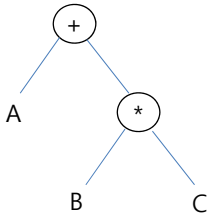
∴ 스택의 top인 7을 돌려줌

중위 표기 수식의 후위 표기 변환

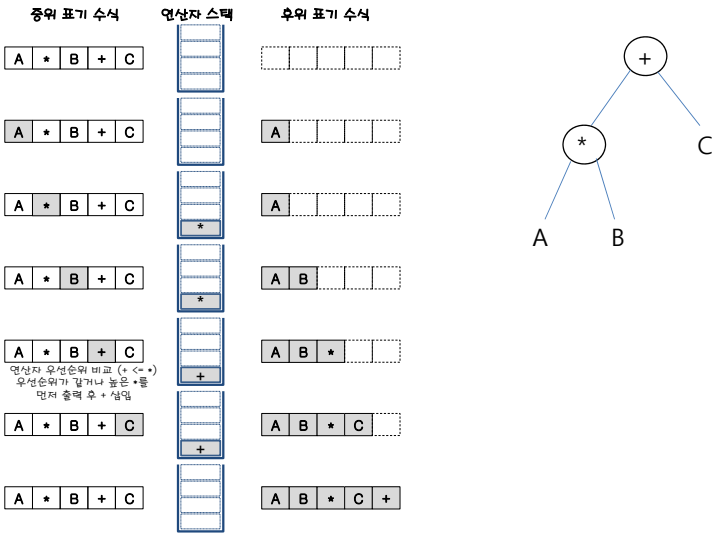
- 중위표기와 후위표기
  - 중위와 후위 표기법의 공통점: 피연산자의 순서가 동일
  - 연산자들의 순서만 다름(우선순위순서)
    - 연산자만 스택에 저장했다가 출력
    - 2+3*4 -> 234*+
- 알고리즘
  - 피연산자를 만나면 그대로 출력
  - 연산자를 만나면 스택에 저장했다가 스택보다 우선 순위가 낮은 연산자가 나오면 그때 출력
  - 왼쪽 괄호는 우선순위가 가장 낮은 연산자로 취급
  - 오른쪽 괄호가 나오면 스택에서 왼쪽 괄호위에 쌓여있는 모든 연산자를 출력

중위 → 후위 표기 변환 예

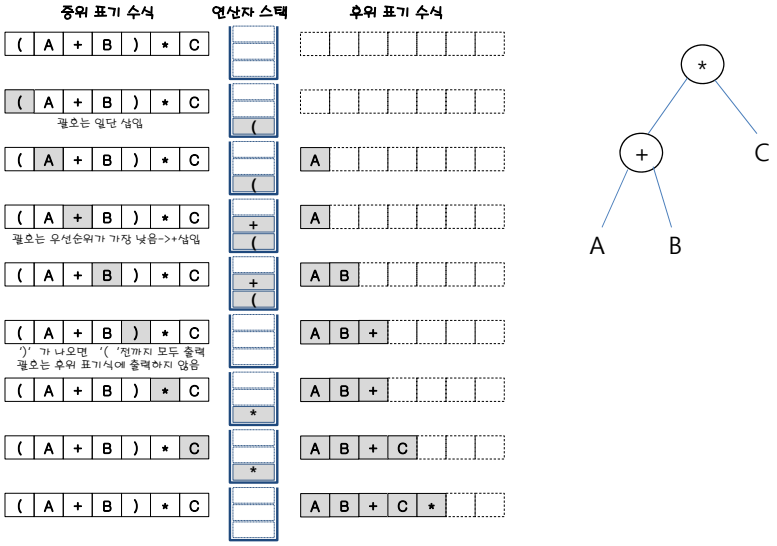




중위 → 후위 표기 변환 예(계속)



후위 표기 변환 예(계속)



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;
```

31

3장 정리

- 스택이란?
  - LIFO 구조
  - push, pop, top, bottom
- 스택 활용
  - 함수호출
  - 연산(후위표기)
- 중위 -> 후위표기 변환
- 스택을 이용한 연산의 적용

32