4장 스택: 배열를 이용한 스택 ADT(교재 프로그램 4.3)

```
#include <stdio.h>
#define MAX_STACK_SIZE 100 // 배열을 이용한 스택은 사이즈의 제한이 있다

// 스택을 위한 타입 정의
typedef int element;

typedef struct {
        element data[MAX_STACK_SIZE];
        int top;
} StackType;

// 스택 초기화 함수
void init(StackType *s)
{
        [          ]
}

// 공백 상태 검출 함수
int is_empty(StackType *s)
{
        [          ]
}

// 포화 상태 검출 함수
int is_full(StackType *s)
{
        [          ]
}

// 삽입함수
void push(StackType *s, element item)
{
        if( is_full(s) ) {
                fprintf(stderr,"스택 포화 에러₩n");
                return;
        }
        else
                [          ]
}
```

S → top = -1;

if (s→top == -1)
    return 1;
else
    return 0;

→ return (s→top == -1);

return (s→top == MAX-1);

S → top++;
S → data[s→top] = item;

S→data[++(s→top)] = item;

```
// 삭제함수
element pop(StackType *s)
{
        if( is_empty(s) ) {
                fprintf(stderr, "스택 공백 에러₩n");
                exit(1);
        }
        else
                [          ]
}

// 피크함수
element peek(StackType *s)
{
        if( is_empty(s) ) {
                fprintf(stderr, "스택 공백 에러₩n");
                exit(1);
        }
        else
                [          ]
}

// 주함수
void main()
{
        StackType s;

        init(&s);
        push(&s, 1);
        push(&s, 2);
        push(&s, 3);
        printf("%d₩n", pop(&s));
        printf("%d₩n", pop(&s));
        printf("%d₩n", pop(&s));
        printf("%d₩n", is_empty(&s));
}
```

return S→data[(s→top)--];

element e;
e = s→data[s→top];
s→top--;
return e;

return S→data[s→top];

1-2

StackType s; 구조체 변수

스택의 응용

Infix_to_postfix(exp) // 중위식을 후위식으로

```
스택 s 를 생성하고 초기화
while (exp 에 처리할 문자가 남아 있으면 do
    ch ← 다음에 처리할 문자
    if ch 가 피연산자이면
        ch 를 출력
    else if '+', '-', '*', '/' 연산자이면
        while (ch 의 우선순위 <= 스택의 가장 위에 있는 것의 우선순위)
            스택에서 꺼내서 출력
        ch 를 스택에 넣는다.
    else if '('이면
        무조건 스택에 넣는다
    else if ')'이면
        왼쪽 괄호를 만날때까지 스택에서 꺼내서 출력(괄호는 출력하지 않는다)

스택에 남은 연산자를 꺼내서 출력
```

eval(postExp) // 후위식의 계산

```
스택 s 를 생성하고 초기화
while (postExp 에 처리할 문자가 남아 있으면 do
    ch ← 다음에 처리할 문자
    if ch 가 피연산자이면
        스택에 넣는다
    else // ch 가 연산자이면
        스택에서 꺼내서 op2
        스택에서 꺼내서 op1
        op1, op2 를 이용하여 연산자로 계산해서 스택에 넣는다.

마지막에 남은 것이 후위식 계산의 결과
```

HW 2 : 스택

■ HW2_0

5장 스택의 Quiz/연습문제 중 일부

■ HW2_1(배열로 구현된 스택)

□ HW2_1_1

배열로 구현된 스택(프로그램 5.3)을 사용하여 다음과 같은 순서로 프로그램을 작성하라.

(1) 현재 스택에 들어 있는 요소들을 다음과 같은 형식으로 출력하는 함수 stack_print(StackType *s)를 작성하라. 예를 들어서 10, 20, 30 순으로 입력된 스택을 출력하면

```
30 <- top
20
10
```

(2) 다음과 같은 순서로 정수를 스택에 삽입, 삭제하도록 <프로그램 5.3>의 main 함수를 변형하라. 삽입, 삭제할 때마다 stack_print() 함수를 호출하여 현재 스택의 모습을 출력하라. 비어있을 경우는 <empty>를 출력한다. 배열의 크기인 MAX_STACK_SIZE 는 3 으로 한다.

10 삽입 -> 20 삽입 -> 30 삽입 -> 40 삽입 -> 삭제 -> 50 삽입 -> 삭제 -> 삭제 -> 삭제

□ HW2_1_2

스택에 다음과 같이 정수와 문자열을 동시에 저장하려고 하면 앞의 프로그램을 어떻게 변경하여야 하는가?(힌트: 프로그램 5.2 처럼 구조체 스택을 사용한다.)

(10, "ten") 삽입 -> (20, "twenty") 삽입 -> (30, "thirty") 삽입 -> (40, "forty") 삽입 -> 삭제 -> (50, "fifty") 삽입 -> 삭제 -> 삭제 -> 삭제

<HW2_1_1 의 실행결과 >

```
C:₩windows₩system32₩cmd.exe
<empty>
--
10 <- top
--
20 <- top
10
--
30 <- top
20
10
--
스택 포화 에러
30 <- top
20
10
--
20 <- top
10
--
50 <- top
20
10
--
20 <- top
10
--
10 <- top
--
<empty>
--
계속하려면 아무 키나 누르십시오 . . .
```

<HW2_1_2 의 실행결과>

```
C:₩windows₩system32₩cmd.ex
<empty>
--
[10, ten] <- top
--
[20, twenty] <- top
[10, ten]
--
[30, thirty] <- top
[20, twenty]
[10, ten]
--
스택 포화 에러
[30, thirty] <- top
[20, twenty]
[10, ten]
--
[20, twenty] <- top
[10, ten]
--
[50, fifty] <- top
[20, twenty]
[10, ten]
--
[20, twenty] <- top
[10, ten]
--
[10, ten] <- top
--
<empty>
--
계속하려면 아무 키나 누르십시오
```


■ HW2_2(연결리스트로 구현된 스택)

□ HW2_2_1

연결 리스트로 구현된 스택(프로그램 5.4)에 대하여 앞의 HW2_1_1 의 (1)과 (2)를 구현하라. 실행결과는 어떤 차이가 있는가?

□ HW2_2_2

연결 리스트로 구현된 스택(프로그램 5.4)을 이용하여 앞의 HW2_1_2 의 문제를 다시 풀라. 실행결과는 어떤 차이가 있는가?

.

■ HW2_3(배열로 구현된 스택과 연결리스트로 구현된 스택의 수행시간 비교)

□ HW2_3_1, HW2_3_2

HW2_2_1(배열 구현)과 HW2_3_1(연결 리스트 구현) 각각의 프로그램에 main 함수를 아래와 같이 대체하여 100,000 번의 삽입과 50,000 번의삭제를. 각각에 대해서 수행시간을 기록하고 비교해보라. 어느 것이 더 빠른가? 그리고 그 이유는?

```
#include <time.h>
int main(void)
{
	StackType s; // HW2_3_2의 경우는 LinkedStackType s;

	time_t start, finish;
	double duration;
	int i;

	init(&s);
	start = clock();

	for(i = 0; i < 100000; i++) {
		push(&s, 10);
		if (i % 2 == 1)
		 pop(&s);
	}
	finish = clock();

	duration = (double)(finish - start) / CLOCKS_PER_SEC;
	printf("%f초 입니다. Wn", duration);
}
```


■ HW2_4(스택의 활용)

4.4

스택의 연산(프로그램 5.3 혹은 프로그램 5.4)를 이용하여 문제를 풀어보는 간단한 프로그램이다. 우리가 배운 스택이 어떻게 활용되는 가를 익혀보기 위한 문제이다.

palindrome 이란 앞뒤 어느 쪽에서 읽어도 같은 역문자열을 의미한다. 예를 들어 “eye”, “abccba” 등이다. 스택을 이용하여 주어진 문자열이 palindrome 인지 아닌지를 결정하는 프로그램을 작성하라.

- 배열이나 연결리스트중 하나를 사용한다.(즉 프로그램 5.3 이나 프로그램 5.4 의 **스택 연산(함수)들을 사용**)
- 알파벳 소문자만 다룬다고 가정하자.
- 주어진 문자열에는 공백문자가 없다고 가정하자. 즉 eye, abccba, madam, abcde 등의 소문자로 이루어진 문자들로 구성된 문자열만 다룬다.

```
int palindrome(char str[])
{
   StackType s; // 배열을 사용할 경우, 연결리스트를 사용하면 LinkedStackType s;

   // 필요한 변수들 선언


   // 스택을 초기화하라


   //str 의 문자들을 스택에 넣는다









   //스택에서 하나씩 빼면서 str 의 문자들과 차례로 비교









}

int main(void)
{
   char word[MAX_STRING];

   printf("Enter a word to check palindrome: ");
   scanf("%s", word);

   if (palindrome(word))
          printf("palindrome 입니다.Wn");
   else
          printf("palindrome 이 아닙니다.Wn");
}
```