Chapter 06.
스택

9주차

스택의 ADT
(Last-In-First-Out (LIFO), 후입선출)

- `void StackInit(Stack * pstack);`
  - 스택의 초기화를 진행한다.
  - 스택 생성 후 제일 먼저 호출되어야 하는 함수이다.

- `int SIsEmpty(Stack * pstack);`
  - 스택이 빈 경우 TRUE(1)을, 그렇지 않은 경우 FALSE(0)을 반환한다.

- `void SPush(Stack * pstack, Data data);`
  - 스택에 데이터를 저장한다. 매개변수 data로 전달된 값을 저장한다.

- `Data SPop(Stack * pstack);`
  - 마지막에 저장된 요소를 삭제한다.
  - 삭제된 데이터는 반환이 된다.
  - 본 함수의 호출을 위해서는 데이터가 하나 이상 존재함이 보장되어야 한다.

- `Data SPeek(Stack * pstack);`
  - 마지막에 저장된 요소를 반환하되 삭제하지 않는다.
  - 본 함수의 호출을 위해서는 데이터가 하나 이상 존재함이 보장되어야 한다.

스택의 배열기반 구현
(구조체 정의)

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

#define TRUE      1
#define FALSE     0
#define STACK_LEN          100

typedef int Data;

typedef struct _arrayStack
{
        Data stackArr[STACK_LEN];
        int topIndex;
} Stack;
```

스택의 배열기반 구현
(초기화, 비었는지 확인)

```
void StackInit(Stack * pstack)
{
	pstack->topIndex = -1;
}

int SIsEmpty(Stack * pstack)
{
	if (pstack->topIndex == -1)
		return TRUE;
	else
		return FALSE;
}
```

스택의 배열기반 구현
(push, pop)

```
void SPush(Stack * pstack, Data data) {
        if (pstack->topIndex >= STACK_LEN - 1)
                exit(-1);
        pstack->topIndex += 1;
        pstack->stackArr[pstack->topIndex] = data;
}

Data SPop(Stack * pstack) {
        int rIdx;

        if (SIsEmpty(pstack))        {
                printf("Stack Memory Error!");
                exit(-1);
        }
        rIdx = pstack->topIndex;
        pstack->topIndex -= 1;
        return pstack->stackArr[rIdx];
}
```

스택의 배열기반 구현 (peek)

```
Data SPeek(Stack * pstack)
{
	if (SIsEmpty(pstack))
	{
		printf("Stack Memory Error!");
		exit(-1);
	}

	return pstack->stackArr[pstack->topIndex];
}
```

스택의 배열기반 구현
(main 함수)

```
int main(void)
{
        Stack stack;
        StackInit(&stack);

        SPush(&stack, 1);   SPush(&stack, 2);
        SPush(&stack, 3);   SPush(&stack, 4);
        SPush(&stack, 5);

        while (!SIsEmpty(&stack))
                printf("%d ", SPop(&stack));

        return 0;
}
```

Stack의 생성 및 초기화

데이터 넣기

데이터 꺼내기

스택의 리스트기반 구현
(구조체 정의)

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>

#define TRUE	1
#define FALSE	0

typedef int Data;
typedef struct _node
{
	Data data;
	struct _node * next;
} Node;

typedef struct _listStack
{
	Node * head;
} Stack;
```

스택의 리스트기반 구현
(초기화, 비었는지 확인, peek)

```
void StackInit(Stack * pstack) {
	pstack->head = NULL;
}

int SIsEmpty(Stack * pstack) {
	if (pstack->head == NULL)
		return TRUE;
	else
		return FALSE;
}

Data SPeek(Stack * pstack) {
	if (SIsEmpty(pstack)) {
		printf("Stack Memory Error!");
		exit(-1);
	}

	return pstack->head->data;
}
```

스택의 리스트기반 구현 (push)

```
void SPush(Stack * pstack, Data data) {
        Node * newNode = (Node*)malloc(sizeof(Node));

        newNode->data = data;
        newNode->next = pstack->head;

        pstack->head = newNode;
}
```

스택의 리스트기반 구현
(pop)

```
Data SPop(Stack * pstack) {
        Data rdata;
        Node * rnode;

        if (SIsEmpty(pstack)) {
                printf("Stack Memory Error!");
                exit(-1);
        }

        rdata = pstack->head->data;
        rnode = pstack->head;

        pstack->head = pstack->head->next;
        free(rnode);

        return rdata;
}
```

스택의 리스트기반 구현
(main 함수 : 배열기반 스택과 동일!)

```
int main(void)
{
        Stack stack;
        StackInit(&stack);

        SPush(&stack, 1);   SPush(&stack, 2);
        SPush(&stack, 3);   SPush(&stack, 4);
        SPush(&stack, 5);

        while (!SIsEmpty(&stack))
                printf("%d ", SPop(&stack));

        return 0;
}
```

Stack의 생성 및 초기화

데이터 넣기

데이터 꺼내기

계산기 프로그램
(계산할 식의 형태)

다음과 같은 문장의 수식을 계산할 수 있어야 한다.

(3 + 4) * (5 / 2) + (7 + (9 – 5))

그러기 위해서는 아래 두 가지를 고려해야 한다.

소괄호 파악 -> 그 부분을 먼저 연산.

연산자의 우선순위 반영.

스택을 이용하여 구현!

계산기 프로그램
(세 가지 수식의 표기법: 전위, 중위, 후위)

중위 표기법(infix notation) **예) 5 + 2 / 7**

수식 내에 연산 순서에 대한 정보가 없다.

-> 소괄호와 연산자의 우선순위를 고려해 줘야 함.

전위 표기법(prefix notation) **예) + 5 / 2 7**

수식 내에 연산의 순서에 대한 정보가 있다.

-> 소괄호와 연산의 우선순위를 고려할 필요 없다.

후위 표기법(postfix notation) **예) 5 2 7 / +**

수식 내에 연산의 순서에 대한 정보가 있다.

-> 소괄호와 연산의 우선순위를 고려할 필요 없다.

계산기 프로그램
(중위표기식 -> 후위표기식으로 전환)

(1) 피 연산자는 그대로 결과로 출력.

(2) 연산자는 스택에 push

- (스택 top 연산자의 우선순위) < (새로 push할 연산자의 우선순위)

  될 때까지 pop하여 결과로 출력.

(3) 식을 모두 처리했으면 스택이 빌 때까지 popup하여 출력.

계산기 프로그램
(중위표기식 -> 후위표기식으로 전환)

▶ [그림 06-3: 수식 변환의 과정 1/7]

▶ [그림 06-5: 수식 변환의 과정 3/7]

▶ [그림 06-7: 수식 변환의 과정 5/7]

▶ [그림 06-9: 수식 변환의 과정 7/7]