7주차 2차시 스택의 구현과 기본응용

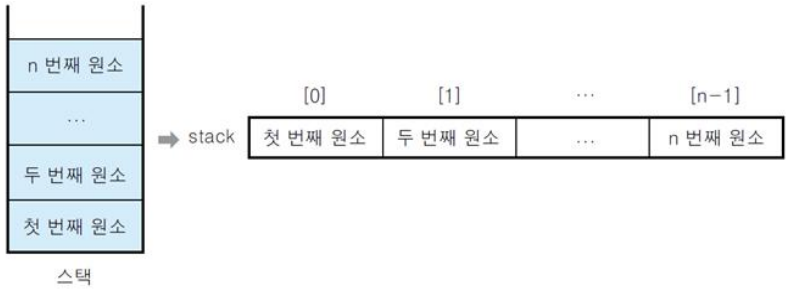
【학습목표】

1. 순차 자료구조 방법을 이용한 스택과 연결 자료구조 방법을 이용한 스택을 구현해 볼 수 있다.
2. 스택의 기본 응용에 대하여 설명할 수 있다.

학습내용1 : 순차 자료구조를 이용한 스택의 구현

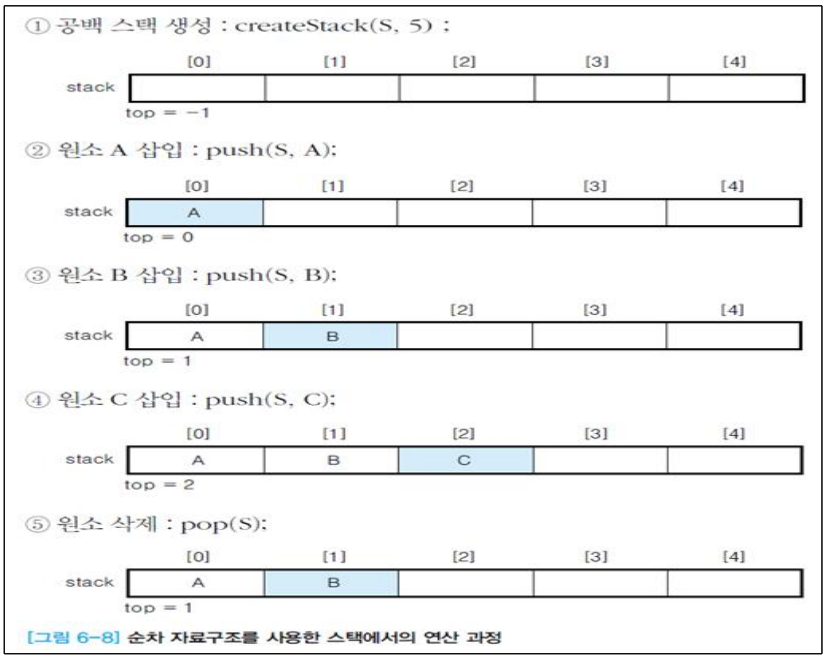
1. 순차 자료구조인 1차원 배열을 이용하여 스택을 구현

- 스택의 크기 : 배열의 크기
- 스택에 쌓이는 순서 : 배열의 인덱스
  - 스택의 첫 번째 원소 : 인덱스 0번
  - 스택의 n번째 원소 : 인덱스 n-1
- 변수 top : 스택에 저장된 마지막 원소에 대한 인덱스 저장
  - 공백 상태 : top = -1(초기값)
  - 포화 상태 : top = n - 1



[그림 6-7] 스택에 대한 1차원 배열 표현

2. 크기가 5인 1차원 배열의 스택 연산 과정



[그림 6-8] 순차 자료구조를 사용한 스택에서의 연산 과정

3. 순차 자료구조를 이용하여 구현한 스택 프로그램

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  #define STACK_SIZE 100
04
05  typedef int element;      // int를 스택 element의 자료형으로 정의
06  element stack[STACK_SIZE];
07  int top= -1;              // 스택의 top의 초기값을-1로 설정
08
09  void push(element item)   // 스택의 삽입 연산
10  {
11          if(top >= STACK_SIZE-1) {    // 스택이 이미 Full인 경우
12             printf("₩n₩n Stack is FULL ! ₩n");
13             return;
14          }
15          else stack[++top]=item;
16   }
17
18  element pop()   // 스택의 삭제 후 반환 연산
19  {
20          if(top==-1) {   // 현재 스택이 공백인 경우
21             printf("₩n₩n Stack is Empty!!₩n");
22             return 0;
23          }
24          else return stack[top--];
25  }
```

```
27  void del()      // 스택의 삭제 연산
28  {
29          if(top==-1) {     // 현재 스택이 공백인 경우
30             printf("₩n₩n Stack is Empty !₩n");
31             exit(1);
32          }
33          else top--;
34  }
35
36  element peek()          // 스택의 top 원소 검색 연산
37  {
38          if(top==-1){     // 현재 스택이 공백인 경우
39             printf("₩n₩n Stack is Empty !₩n");
40             exit(1);
41          }
42          else return stack[top];
43  }
44
45  void printStack()        // 스택 내용 출력 연산
46  {
47          int i;
48          printf("₩n STACK [ ");
49          for(i=0; i<=top; i++)
50             printf("%d ",stack[i]);
51          printf("] ");
52   }
53
```

```
54  void main(void)
55  {
56          int item;
57          printStack();
58          push(1);
59          printStack();
60          push(2);
61          printStack();
62          push(3);
63          printStack();
64
65          item = peek();
66          printStack();
67          printf("peek top => %d", item);
68
69          del();
70          printStack();
71
```

```
72          item = pop();
73          printStack();
74          printf("₩t pop top => %d", item);
75
76          item = pop();
77          printStack();
78          printf("₩t pop top => %d", item);
79
80           pop();
81
82          getchar();
83  }
```

4. 실행결과

```
C:₩자료구조-예제₩2부₩6장₩Debug₩예제6-1.exe
STACK [ ]
STACK [ 1 ]
STACK [ 1 2 ]
STACK [ 1 2 3 ]
STACK [ 1 2 3 ] peek top => 3
STACK [ 1 2 ]
STACK [ 1 ]     pop top => 2
STACK [ ]       pop top => 1

Stack is Empty!!
```

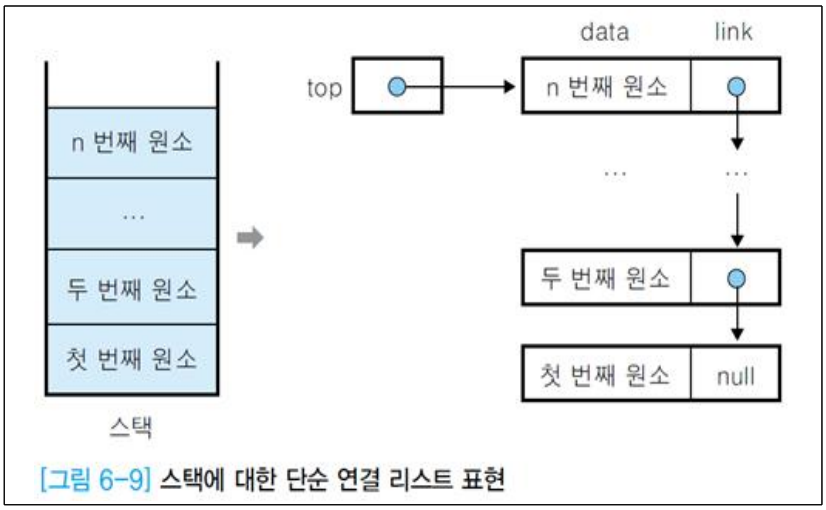
- 순차 자료구조를 이용한 스택은 구현이 용이
- 고정된 배열을 사용하기 때문에 스택의 크기를 변경하기 어려움
- 메모리 낭비가 생길 수 있음

학습내용2 : 연결 자료구조를 이용한 스택의 구현

1. 단순 연결 리스트를 이용하여 스택을 구현

* 스택의 원소 : 단순 연결 리스트의 노드
- 스택 원소의 순서 : 연결 리스트 노드의 링크로 표현
- push : 리스트의 마지막에 노드 삽입
- pop : 리스트의 마지막 노드 삭제

* 변수 top : 단순 연결 리스트의 마지막 노드를 가리키는 포인터 변수
- 초기 상태 : top = null

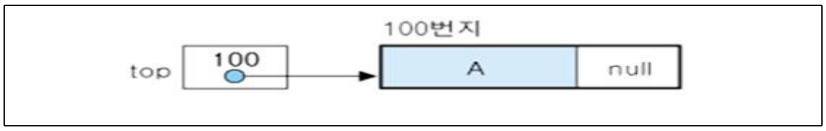


[그림 6-9] 스택에 대한 단순 연결 리스트 표현

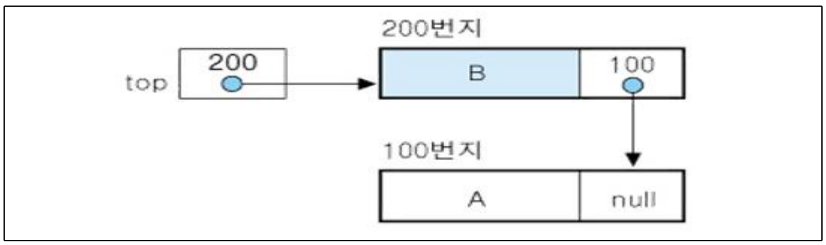
2. 단순 연결 리스트를 사용한 스택에서의 연산 과정

① 공백 스택 생성 : createStack(S);

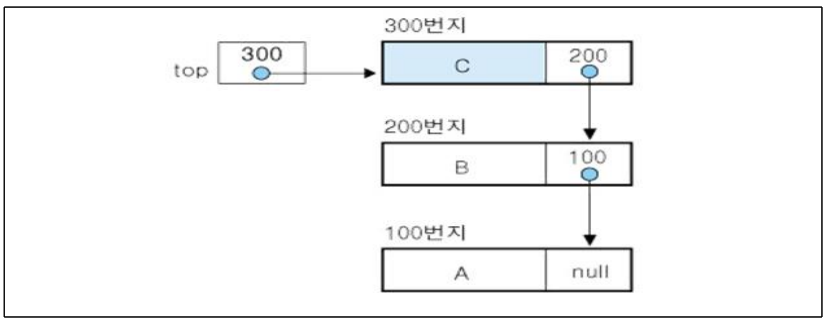
② 원소 A 삽입 : push(S, A);



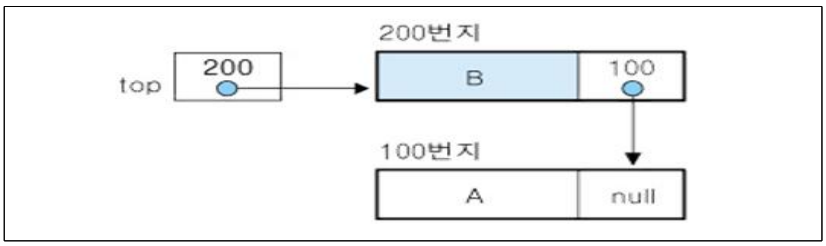
③ 원소 B 삽입 : push(S, B);



④ 원소 C 삽입 : push(S, C);



⑤ 원소 삭제 : pop(S, C);



3. 단순 연결 리스트를 이용하여 구현한 스택 프로그램

```
001  #include <stdio.h>
002  #include <stdlib.h>
003  #include <string.h>
004
005  typedef int element;    // int를 스택 element의 자료형으로 정의
006
007  typedef struct stackNode {   // 스택의 노드 구조 정의
008        element data;
009        struct stackNode *link;
010  }stackNode;
011
012  stackNode* top;     // 스택의top 노드를 지정하기 위한 포인터 top 선언
013
014  void push(element item)    // 스택 삽입 연산
015  {
016        stackNode* temp=(stackNode *)malloc(sizeof(stackNode));
017        temp->data = item;
018        temp->link = top;
019        top = temp;
020  }
021
```

```
022  element pop()    // 스택의 삭제 후 반환 연산
023  {
024        element item;
025        stackNode* temp=top;
026
027        if(top == NULL) {     // 현재 스택이 공백 리스트인 경우
028           printf("₩n₩n Stack is empty !₩n");
029           return 0;
030        }
031        else{                  // 현재 스택이 공백 리스트가 아닌 경우
032           item = temp->data;
033           top = temp->link;
034           free(temp);
035           return item;
036        }
037  }
038
```

```
052  void del()      // 스택의 삭제 연산
053  {
054        stackNode* temp;
055        if(top == NULL) {    // 현재 스택이 공백 리스트인 경우
056           printf("₩n₩n Stack is empty !₩n");
057        }
058        else {                 // 현재 스택이 공백 리스트가 아닌 경우
059           temp = top;
060           top = top->link;
061           free(temp);
062        }
063  }
064
065  void printStack()         // 스택의 내용 출력 연산
066  {
067        stackNode* p=top;
068        printf("₩n STACK [ ");
069        while(p){
070           printf("%d ",p->data);
071           p = p->link;
072        }
073        printf("] ");
074  }
075
```

```
076  void main(void)
077  {
078       element item;
079       top = NULL;
080       printStack();
081       push(1);
082       printStack();
083       push(2);
084       printStack();
085       push(3);
086       printStack();
087
088       item = peek();
089       printStack();
090       printf("peek top => %d", item);
091
092       del();
093       printStack();
094
```

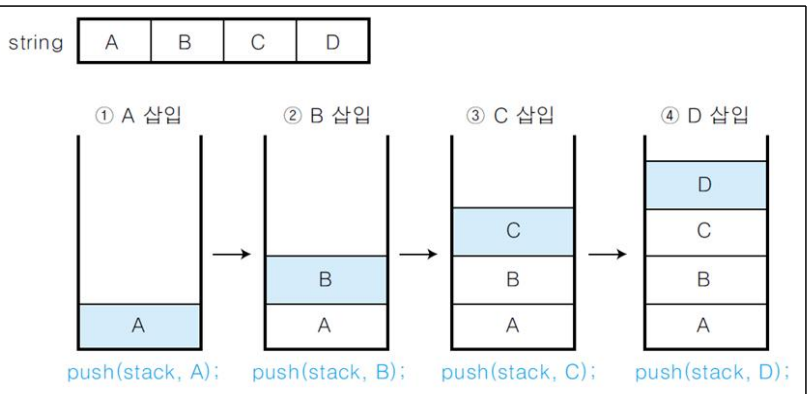
4. 실행결과

```
C:₩자료구조-예제₩2부₩6장₩Debug₩예제6-2.exe

STACK [ ]
STACK [ 1 ]
STACK [ 2 1 ]
STACK [ 3 2 1 ]
STACK [ 3 2 1 ] peek top => 3
STACK [ 2 1 ]
STACK [ 1 ]     pop top => 2
STACK [ ]       pop top => 1

Stack is empty !
```

학습내용3 : 역순문자열 만들기

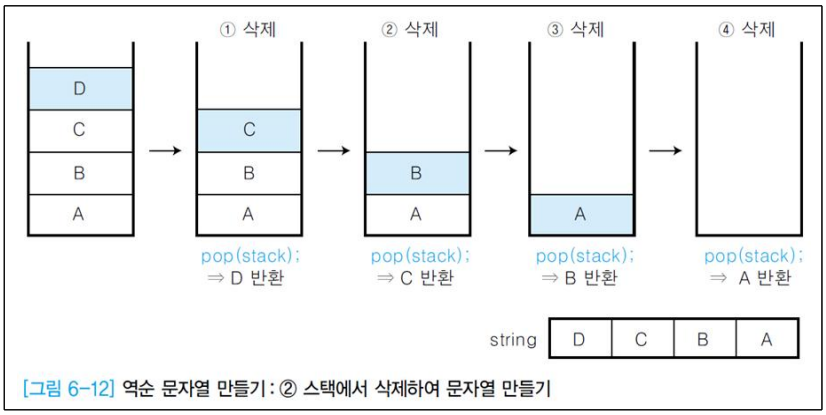
1. 스택의 LIFO 성질을 이용하여 역순 문자열 만들기

① 문자열을 순서대로 스택에 삽입



[그림 6-11] 역순 문자열 만들기 : ① 순서대로 스택에 삽입하기

② 스택을 pop하여 문자열로 저장하기



[그림 6-12] 역순 문자열 만들기 : ② 스택에서 삭제하여 문자열 만들기

【학습정리】

1. 순차 자료구조를 이용하여 스택을 구현하기 위하여 배열을 사용한다.
- 1차원 배열 stack[n]을 사용할 때 n은 배열 크기로서 배열 원소의 개수를 나타내는데, 이것은 스택의 크기가 되고 배열의 인덱스는 스택에 원소가 쌓이는 순서가 된다.

2. 연결 자료구조를 이용하여 스택을 구현하려면 스택의 원소는 연결 리스트의 노드가 된다.
- 스택에 원소를 삽입할 때마다 연결 리스트에 노드를 하나씩 할당하여 연결한다. 스택 원소의 순서는 연결 리스트 노드의 링크를 사용하여 표현하고 스택의 top은 마지막 원소 노드에 대한 포인터 top을 사용한다.