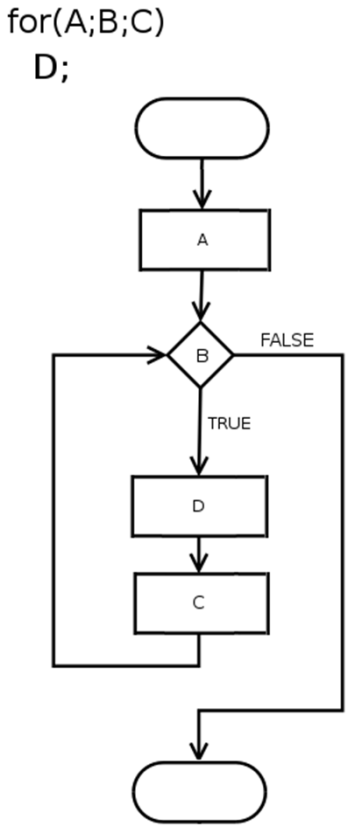
Clickseo.com

자료구조 및 알고리즘

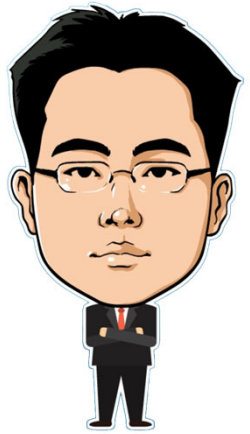


스택
(Stack)

Seo, Doo-Ok

Clickseo.com

clickseo@gmail.com



목 차

백문이불여일타(百聞而不如一打)

- 스택의 이해
- 스택 구현

스택의 이해

- **스택의 이해**

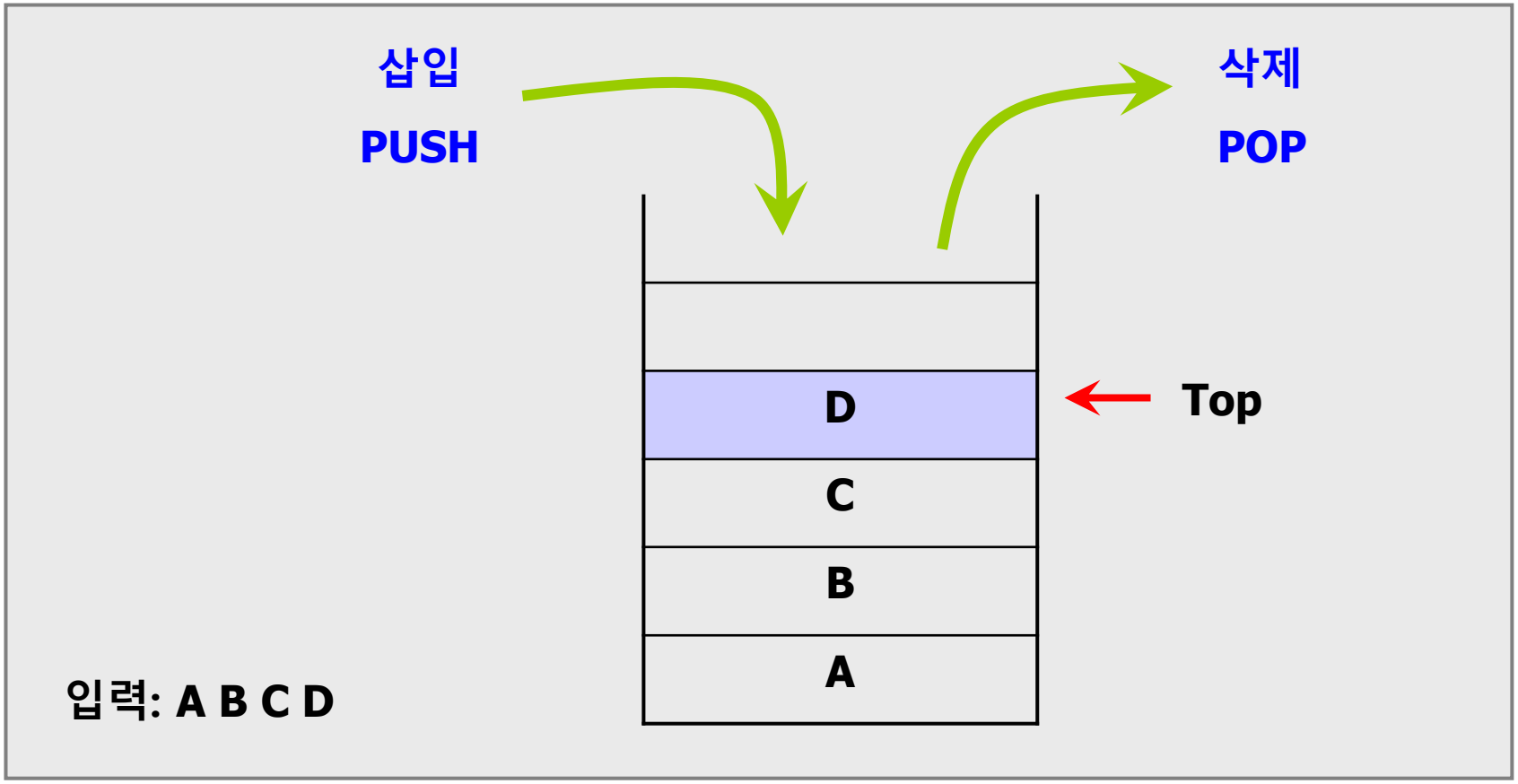
백문이불여일타(百聞而不如一打)

  - Python 내장 클래스: list
  - C++ STL: <stack> 클래스
  - 다양한 스택 활용
  - 스택 구현: 순차.연결 자료구조

- **스택 구현**

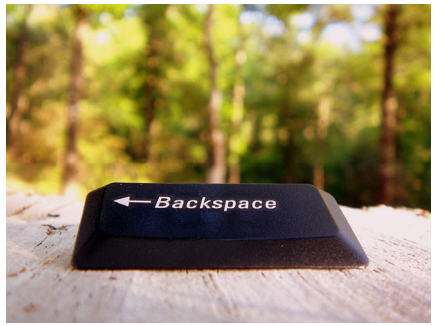
스택의 이해 (1/5)

- **스택**(Stack)
  - 후입선출(**LIFO**, Last-In-First-Out)



스택의 이해 (2/5)

- **다양한 스택 활용**
  - 백스페이스(Backspace) 키: 최근에 입력한 글자 삭제
  - 최근에 작업한 순으로 취소: Ctrl + Z
  - 역순 문자열, 진법 변환



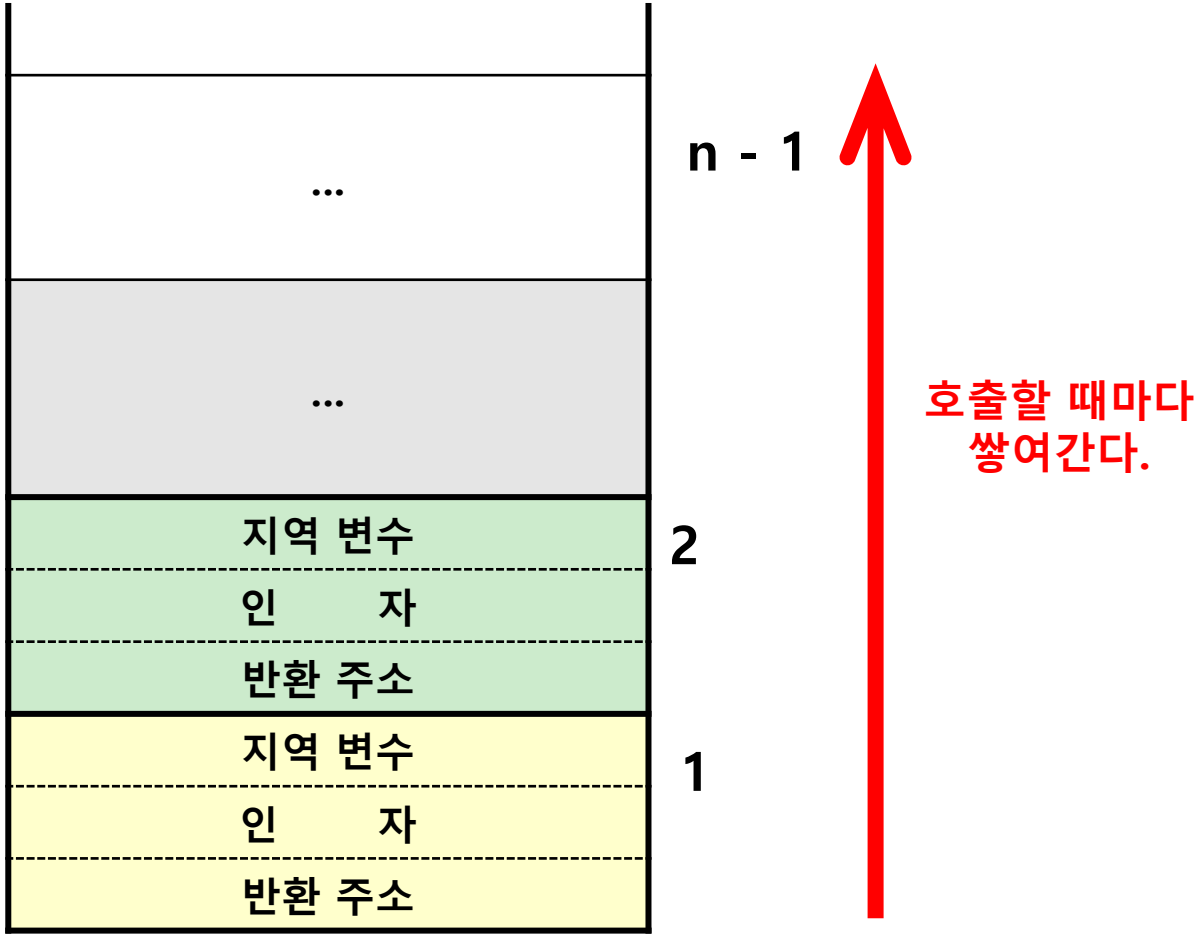
  - 후위 표기법(Postfix Notation)
    - 역폴란드 표기법(RPN, Reverse Polish Notation)
    - 수식의 괄호 검사
    - 후위 표기법 변환과 수식 연산

3 4 +

  - 시스템 스택(System Stack)
    - 함수의 호출과 복귀 순서를 스택의 **LIFO** 구조를 응용하여 관리한다.

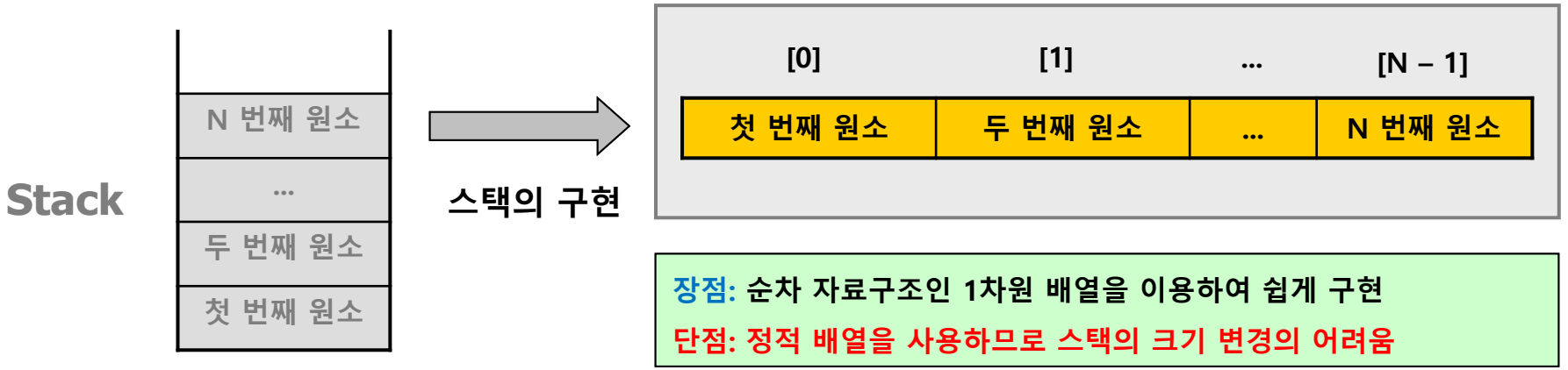
스택의 이해 (3/5)

- 함수 호출 시 동작 과정: 스택 구조



스택의 이해 (4/5)

- **스택 구현:** 순차 자료구조
  - 순차 자료구조인 1차원 배열을 이용하여 구현
    - **스택 크기:** 배열의 크기
    - **스택에 저장된 원소의 순서:** 배열 원소의 첨자
    - 변수 top : 스택에 저장된 마지막 원소에 대한 첨자
      - 공백 상태: top = -1 (초기값)
      - 포화 상태: top = N - 1

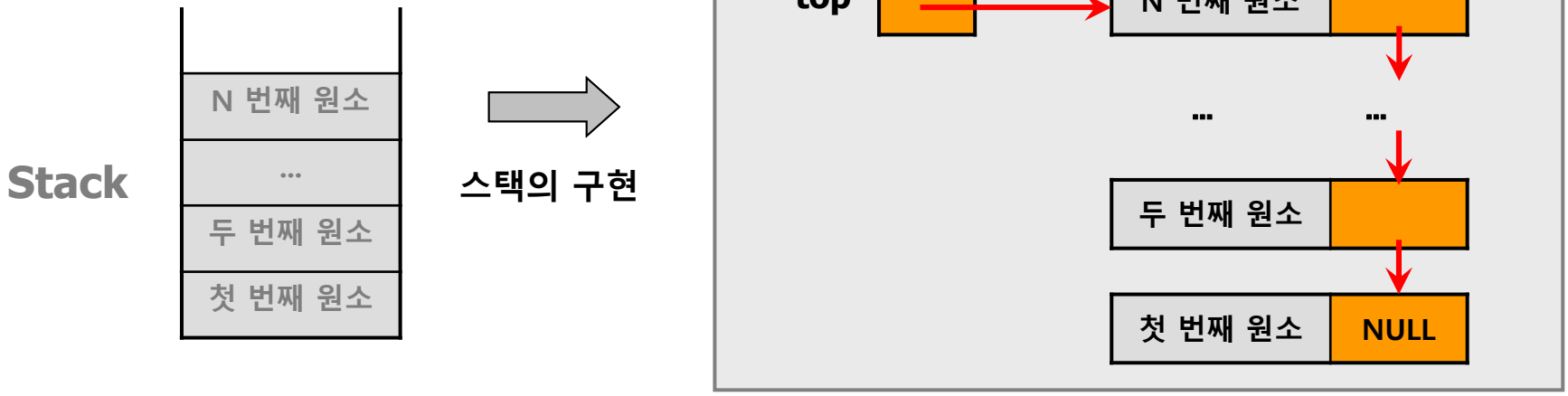


장점: 순차 자료구조인 1차원 배열을 이용하여 쉽게 구현

단점: 정적 배열을 사용하므로 스택의 크기 변경의 어려움

스택의 이해 (5/5)

- **스택 구현:** 연결 자료구조
  - 단순 연결 리스트를 이용하여 구현
    - 스택의 원소: 단순 연결 리스트의 노드
      - **스택 원소의 순서:** 노드의 링크 필드로 연결
      - **push :** 항상 리스트의 첫 번째 노드로 삽입
      - **pop :** 항상 리스트의 마지막 노드를 삭제
    - 변수 top : 단순 연결 리스트의 마지막 노드를 가리키는 포인터형 변수
      - 초기 상태: top = NULL



스택의 이해

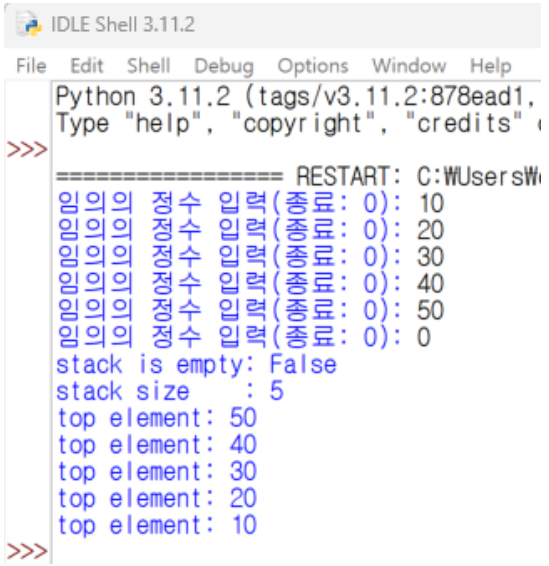
Python 내장 클래스: list

C++ STL: stack 클래스

Python 내장 클래스: list

예제 5-1: 스택의 이해 -- 내장 클래스(list class) | Python

```
s = []

while True:
    num = int(input('임의의 정수 입력(종료: 0): '))
    if num == 0:
        break
    s.append(num)

print(f'stack size     : {len(s)}')
print(f'stack is empty: {len(s) == 0}')

while s:
    # print(f'top element: {s.pop()}')
    print(f'top element: {s[-1]}')
    s.pop()
```



C++ STL: stack 클래스 (1/3)

- **컨테이너 라이브러리(Containers Library)**
  - 컨테이너 어댑터(Container Adaptor)
    - <stack> : 스택(Stack) 구조, **LIFO**(Last in First out) 데이터 구조
    - <queue> : 큐(Queue) 구조, FIFO(First in First out) 데이터 구조
    - <priority_queue> : 우선순위 큐(Priority queue)
    - <flat_set> (since C++23)
    - <flat_map> (since C++23)
    - <flat_multiset> (since C++23)
    - <flat_multimap> (since C++23)

C++ STL: stack 클래스 (2/3)

- **<stack> 클래스**
  - 스택, LIFO(Last in First out)

```
// C++ STL : <stack>
#include <stack>
using namespace std;

stack<DataType> stackName                          // 빈 스택 생성

void          push(const value_type &val);         // 데이터 삽입
void          pop();                               // 데이터 삭제
value_type    &top();                              // 스택의 맨 위의 데이터 반환
Bool          empty() const;                       // 빈 스택 여부
size_type     size() const;                        // 스택의 원소 개수
swap(stack1, stack2)                               // 스택 SWAP
```

C++ STL: stack 클래스 (3/3)

예제 5-2: 스택의 이해 -- C++ STL(stack class) | C++

```
#include <iostream>
#include <stack>
using namespace std;

int main(void)
{
        int             num;
        stack<int>      s;

        while (true) {
                cout << "임의의 정수 입력(종료: 0): ";
                cin >> num;
                if (num == 0)
                        break;
                s.push(num);
        }

        cout << endl;
        cout << "stack is empty: " << s.empty() << endl;
        cout << "stack size    : " << s.size() << endl;

        while (!s.empty()) {
                cout << "top element: " << s.top() << endl;
                s.pop();
        }
        return 0;
}
```

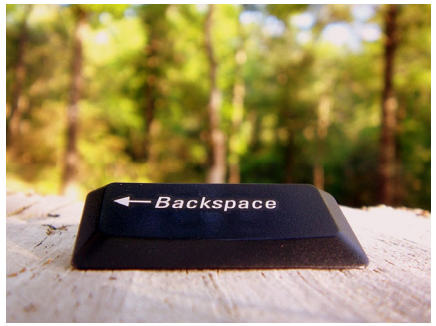
```
Microsoft Visual Studio 디버그
임의의 정수 입력(종료: 0): 10
임의의 정수 입력(종료: 0): 20
임의의 정수 입력(종료: 0): 30
임의의 정수 입력(종료: 0): 40
임의의 정수 입력(종료: 0): 50
임의의 정수 입력(종료: 0): 0

stack is empty: 0
stack size    : 5
top element: 50
top element: 40
top element: 30
top element: 20
top element: 10

C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

스택의 이해

다양한 스택 활용

다양한 스택 활용 (1/13)

- **다양한 스택 활용**
  - 백스페이스(Backspace) 키: 최근에 입력한 글자 삭제
  - 최근에 작업한 순으로 취소: Ctrl + Z
  - 역순 문자열, 진법 변환



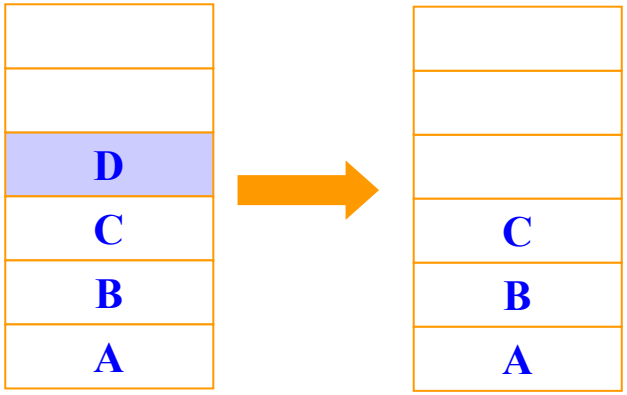
  - 후위 표기법(Postfix Notation)
    - 역폴란드 표기법(RPN, Reverse Polish Notation)
    - 수식의 괄호 검사
    - 후위 표기법 변환과 수식 연산

34+

  - 시스템 스택(System Stack)
    - 함수의 호출과 복귀 순서를 스택의 **LIFO** 구조를 응용하여 관리한다.

다양한 스택 활용 (2/13)

- 다양한 스택 활용
  - 문자열 역순 출력



push(stack, D); pop(stack);

다양한 스택 활용 (3/13)

예제 5-3: 문자열 역순 출력 -- 내장 클래스(list class) | Python

```
s = []

tStr = input('문자열 입력: ')
for ch in tStr:
    s.append(ch)

# 문자열 역순 출력
print('문자열 역순 출력: ', end= '')
while len(s):
    print(f'{s[-1]}', end= '')
    s.pop()
```

다양한 스택 활용 (4/13)

예제 5-3: 문자열 역순 출력 -- C++ STL(stack class) | C++

```
#include <iostream>
#include <string>
#include <stack>
using namespace std;

int main(void)
{
        string          str;
        stack<char>     s;

        cout << "문자열 입력: ";
        getline(cin, str);

        for (char ch : str)
                s.push(ch);
        // for(int i = 0; i < str.length(); ++i)
        //      s.push(str[i]);

        // 문자열 역순 출력
        cout << "문자열 역순 출력: ";
        while (!s.empty()) {
                cout << s.top();
                s.pop();
        }
        cout << endl;
        return 0;
}
```

```
Microsoft Visual Studio 디버그
문자열 입력: ABCD
문자열 역순 출력: DCBA

C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

다양한 스택 활용 (5/13)

예제 5-3: 문자열 역순 출력 -- C++ STL(vector class) | C++

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main(void)
{
        string          str;
        vector<char>    v;

        cout << "문자열 입력: ";
        getline(cin, str);

        for (char ch : str)
                v.push_back(ch);
        // for(int i = 0; i < str.length(); ++i)
        //      v.push_back(str[i]);

        // 문자열 역순 출력
        cout << "문자열 역순 출력: ";
        while (!v.empty()) {
                cout << v.back();
                v.pop_back();
        }
        cout << endl;
        return 0;
}
```

```
Microsoft Visual Studio 디버그
문자열 입력: ABCD
문자열 역순 출력: DCBA

C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

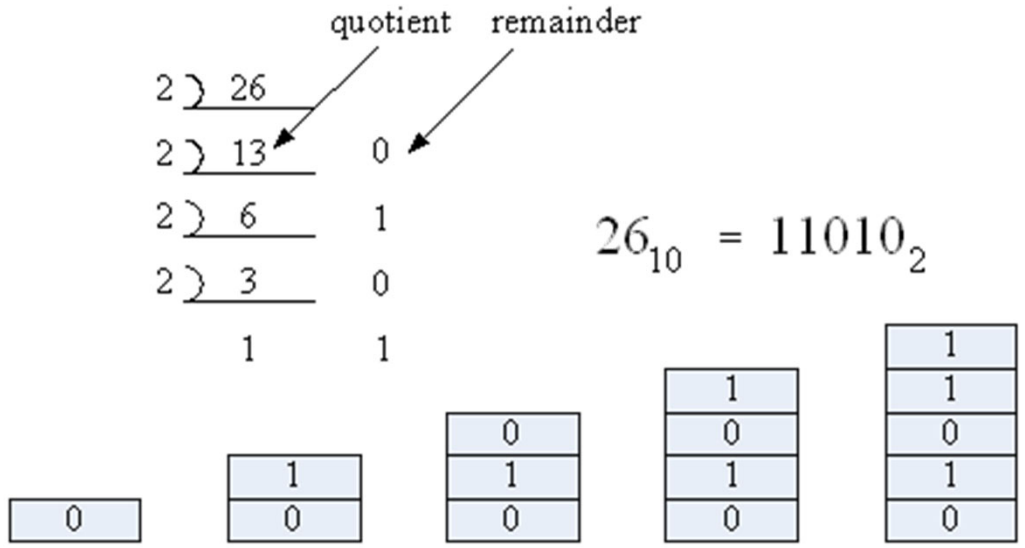
다양한 스택 활용 (6/13)

예제 5-3: 문자열 역순 출력 | C

```c
#include <stdio.h>
#include "LinkedStack.h"      // ListStack, stackNode
#define bufferMAXSIZE  1024

int main(void)
{
        char            str[bufferMAXSIZE];
        LinkedStack     *s = stackCreate();

        printf("문자열 입력: ");
        gets_s(str, sizeof(str));

        char    *pStr = str;
        while (*pStr)
                push(s, *pStr++);

        printf("\n문자열 역순 출력: ");
        while (!stackEmpty(s)) {
                printf("%c", top(s));
                pop(s);
        }
        printf("\n");

        // stackDestroy(s);
        return 0;
}
```

다양한 스택 활용 (7/13)

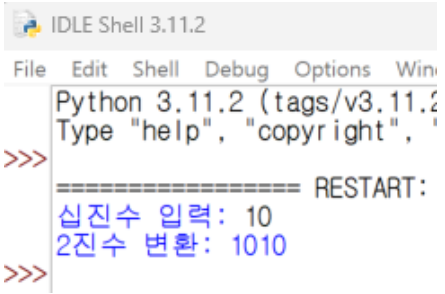
- **다양한 스택 활용**
  - 십진수를 이진수로 변환

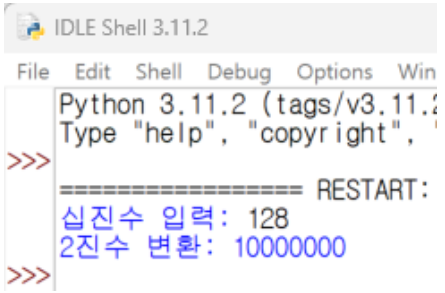


다양한 스택 활용 (8/13)

예제 5-4: 십진수를 이진수로 변환 -- 내장 클래스(list class) | Python

```
s = []

num = int(input('십진수 입력: '))
# 십진수를 이진수로 변환
while num:
    s.append(num%2)
    num//=2

# 이진수 출력
print('2진수 변환: ', end= '')
while len(s):
    print(f'{s[-1]}', end= '')
    s.pop()
```

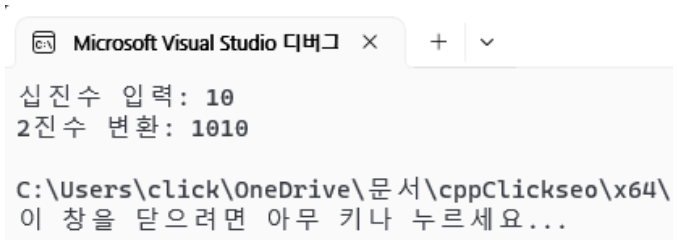


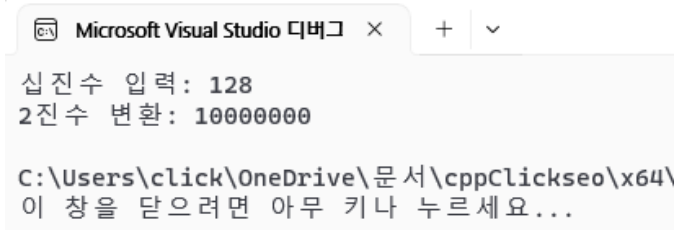


다양한 스택 활용 (9/13)

예제 5-4: 십진수를 이진수로 변환 -- C++ STL(stack class) | C++

```
#include <iostream>
#include <stack>
using namespace std;

int main(void)
{
        int             num;
        stack<int>      s;

        cout << "십진수 입력: ";
        cin >> num;

        // 십진수를 이진수로 변환
        while (num) {
                s.push(num % 2);
                num /= 2;
        }

        // 이진수 출력
        cout << "2이진수 변환: ";
        while (!s.empty()) {
                cout << s.top();
                s.pop();
        }
        cout << endl;
        return 0;
}
```





다양한 스택 활용 (10/13)

예제 5-4: 십진수를 이진수로 변환 -- C++ STL(vector class) | C++

```
#include <iostream>
#include <vector>
using namespace std;

int main(void)
{
        int             num;
        vector<int>     v;

        cout << "십진수 입력: ";
        cin >> num;

        // 십진수를 이진수로 변환
        while (num) {
                v.push_back(num % 2);
                num /= 2;
        }

        // 이진수 출력
        cout << "2진수 변환: ";
        while (!v.empty()) {
                cout << v.back();
                v.pop_back();
        }
        cout << endl;
        return 0;
}
```

```
Microsoft Visual Studio 디버그
십진수 입력: 10
2진수 변환: 1010

C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

```
Microsoft Visual Studio 디버그
십진수 입력: 128
2진수 변환: 10000000

C:\Users\click\OneDrive\문서\cppClickseo\x64\
이 창을 닫으려면 아무 키나 누르세요...
```

다양한 스택 활용 (11/13)

예제 5-4: 십진수를 이진수로 변환 | C

```c
#include <stdio.h>
#include "LinkedStack.h"      // ListStack, stackNode

int main(void)
{
        int             num;
        LinkedStack     *s = stackCreate();

        printf("십진수 입력: ");
        scanf_s("%d", &num);
        // scanf("%d", &num);

        while (num) {
                push(s, num % 2);
                num /= 2;
        }

        printf("\n2진수 변환: ");
        while (!stackEmpty(s)) {
                printf("%d", top(s));
                pop(s);
        }
        printf("\n");

        // stackDestroy(s);
        return 0;
}
```

```
Microsoft Visual Studio 디버그
십진수 입력: 10

2진수 변환: 1010
```

다양한 스택 활용 (12/13)

- **후위 표기법 변환:** 알고리즘

```
infixToPostfix(exp)  // 후위 표기법 변환 알고리즘
    while (true) do
    {
        symbol ← getSymbol(exp);
        case
        {
            symbol = operand : print(symbol);      // 피연산자는 그냥 출력
            symbol = operator :    // 낮은 우선 순위의 연산자를 만날 때 까지 pop
                while (op(Stack[top]) >= op(symbol)) do
                    print(pop(Stack));
                push(Stack, symbol);      // 자신을 push
            symbol = "(" : push(Stack, symbol);    // 스택에 push
            symbol = ")" :  // ')'를 만나면 '('가 나올 때까지 스택에서 pop 하여 출력
                while (Stack[top] ≠ "(" ) do
                    print(pop(Stack));
                pop(Stack);
            symbol = NULL : return;
        }
    }
end infixToPostfix()
```

(3 * 5) – (6 / 2)

>> 3 5 * 6 2 / -

다양한 스택 활용 (13/13)

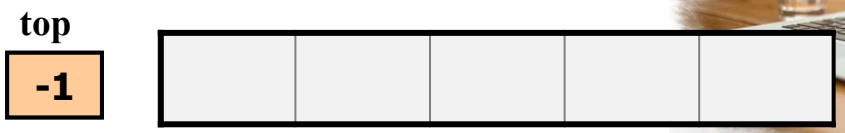
- **후위 표기법 연산:** 알고리즘

```
evalPostfix(exp)    // 후위 표기식의 연산 알고리즘
    while (true) do
    {
        symbol ← getSymbol(exp);
        case
        {
            symbol = operand : push(Stack, symbol);
            symbol = operator :
                operand2 ← pop(Stack));
                operand1 ← pop(Stack));
                // 스택에서 꺼낸 피연산자들을 연산자로 연산
                res ← operand1  op(symbol)  operand2;
                push(Stack, res);          // 결과값을 스택에 push
            symbol = NULL : return;   // 후위 수식의 끝: 최종 결과 값 pop
        }
    }
end evalPostfix()
```

3 5 * 6 2 / - >> (결과) 12

스택의 이해

스택 구현: 순차 자료구조



스택 구현(Python): 순차 자료 구조 (1/3)

- **스택 구현:** 순차 자료구조

```
# 클래스 설계: ArrayStack
class ArrayStack:
    def __init__(self):          # 빈 스택 생성
        self.__stack = []

    def __del__(self):           # 스택 삭제

    def empty(self) -> bool:                    # 빈 스택 여부
    def size(self) -> int:                      # 스택의 원소 개수
    def push(self, num) -> None:                # 데이터 삽입
    def pop(self) -> None:                      # 데이터 삭제
    def top(self):                  # 스택에서 맨 위의 데이터 반환
    def printStack(self) -> None:   # 스택의 전체 데이터 출력
```

스택 구현(C++): 순차 자료 구조 (2/3)

- **스택 구현:** 순차 자료구조

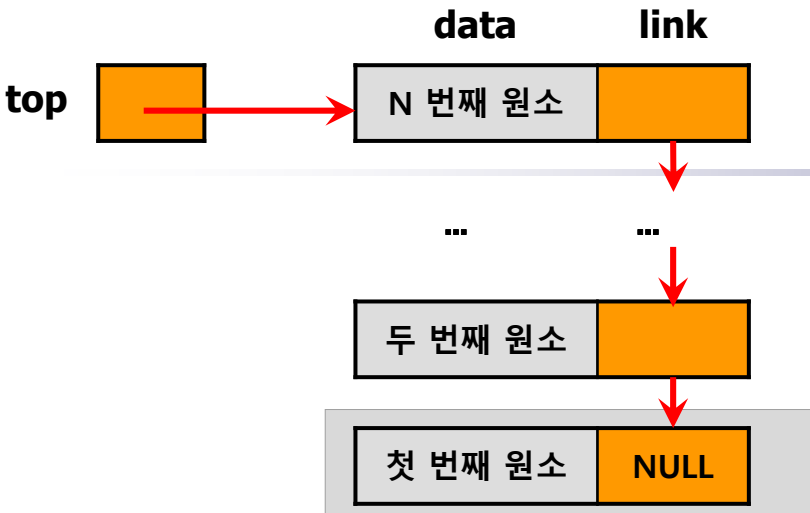
```
// 클래스 설계: ArrayStack
template <typename T>
class ArrayStack {
public:
    ArrayStack(int size = 10);          // 빈 스택 생성
    ~ArrayStack(void);                  // 스택 삭제
    bool    empty(void) const;          // 빈 스택 여부
    bool    full(void) const;           // 포화 상태 여부
    int     size(void) const;           // 원소 개수
    void    push(const T &data);        // 데이터 삽입
    void    pop(void);                  // 데이터 삭제
    T       top(void) const;            // 맨 위의 데이터 반환
    void    printStack(void) const;     // 전체 데이터 출력
private:
    int     top_;
    int     maxSize_;
    T       *stack_;
};
```

스택 구현(C): 순차 자료 구조 (2/3)

- **스택 구현:** 순차 자료구조

```
// #pragma once
#define                 StackMAXSIZE 1024
typedef     int         element;

#ifndef __ArrayStack_H__
#define __ArrayStack_H__

// 구조체 설계: ArrayStack
typedef struct _ArrayStack {
    int                 top;
    element             stack[StackMAXSIZE];
} ArrayStack;

#endif
```


```
// ArrayStack: 스택 생성 및 조작 함수
ArrayStack  *stackCreate(void);                     // 빈 스택 생성
void        stackDestroy(ArrayStack *Stack);        // 스택 삭제
_Bool       stackEmpty(ArrayStack *Stack);          // 빈 스택 여부
_Bool       stackFull(ArrayStack *Stack);           // 스택의 포화 상태 여부
int         stackSize(ArrayStack *Stack);           // 스택의 원소 개수
void        push(ArrayStack *Stack, element data);  // 데이터 삽입
void        pop(ArrayStack *Stack);                 // 데이터 삭제
element     top(ArrayStack *Stack);                 // 스택에서 맨 위의 데이터 반환
void        printStack(ArrayStack *Stack);          // 스택의 전체 데이터 출력
```



스택의 이해

스택 구현: 연결 자료구조

스택 구현(Python): 연결 자료 구조 (1/3)

- **스택 구현:** 연결 자료구조

```
# 클래스 설계: LinkedStack
class LinkedStack:
    class SNode:
        def __init__(self, data, link=None):
            self.data = data
            self.link = link

    # 빈 스택 생성
    def __init__(self):
        self.__top = None
        self.__count = 0

    def __del__(self):                          # 스택 삭제
    def empty(self) -> bool:                    # 빈 스택 여부
    def size(self) -> int:                      # 스택의 원소 개수
    def push(self, data) -> None:               # 데이터 삽입
    def pop(self) -> None:                      # 데이터 삭제
    def top(self):                              # 스택에서 맨 위의 데이터 반환
    def printStack(self) -> None:               # 스택의 전체 데이터 출력
```

스택 구현(C++): 연결 자료 구조 (2/3)

- **스택 구현:** 연결 자료구조

```
// LinkedQueue.cpp
#include "LinkedNode(template).cpp"

// #pragma once
#ifndef __LinkedStack_Template_H__
#define __LinkedStack_Template_H__

// 클래스 설계: LinkedStack
template <typename T>
class LinkedStack {
        LinkedStack();                          // 빈 스택 생성
        ~LinkedStack();                         // 스택 삭제
        bool    empty(void) const;              // 빈 스택 여부
        int     size(void) const;               // 원소 개수
        void    push(const T &data);            // 데이터 삽입
        void    pop(void);                      // 데이터 삭제
        T       top(void) const;                // 맨 위의 데이터 반환
        void    printStack(void) const;         // 전체 데이터 출력
private:
        SNode<T>            *top_;
        int                 count_;
};

#endif
```

```
// LinkedNode(template).cpp
// #pragma once
#ifndef __SNode_Template_H__
#define __SNode_Template_H__

template <typename T>    class LinkedQueue;

// 클래스 설계: SNode
template <typename T>
class SNode {
public:
        SNode(const T &data);
        T       getData(void) const;
private:
        T                   data_;
        SNode<T>            *link_;
        template <typename T> friend class LinkedQueue;
};

#endif
```

스택 구현(C): 연결 자료 구조 (3/3)

- **스택 구현:** 연결 자료구조

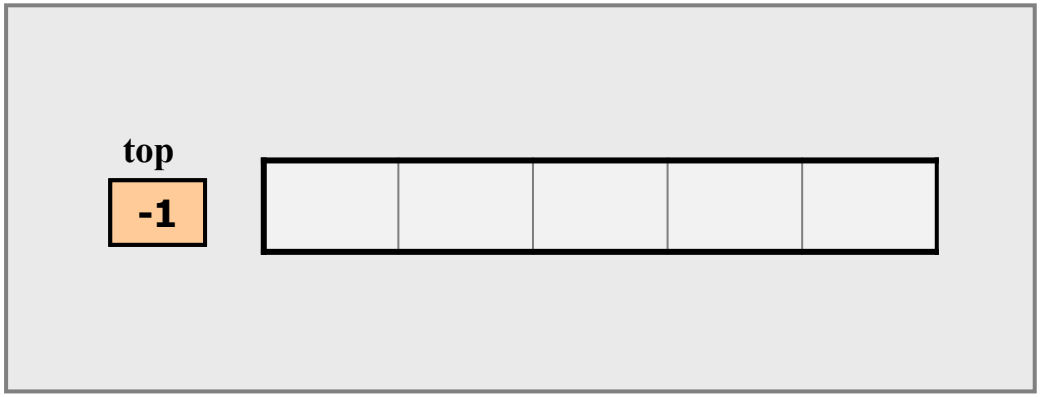
```
// #pragma once
#include "LinkedNode.h"              // SNode, makeSNode

#ifndef __LinkedStack_H__
#define __LinkedStack_H__

// 구조체 설계: LinkedStack
typedef struct _LinkedStack {
        SNode       *top;
        int         count;
} LinkedStack;

#endif
```


```
// LinkedStack: 스택 생성 및 조작 함수
LinkedStack *stackCreate(void);                          // 빈 스택 생성
void        stackDestroy(LinkedStack *Stack);            // 스택 삭제: 모든 노드 삭제
_Bool       stackEmpty(LinkedStack *Stack);              // 빈 스택 여부
int         stackSize(LinkedStack *Stack);               // 스택의 원소 개수
void        push(LinkedStack *Stack, element data);      // 데이터 삽입
void        pop(LinkedStack *Stack);                     // 데이터 삭제
element     top(LinkedStack *Stack);          // 스택에서 맨 위의 데이터 반환
void        printStack(LinkedStack *Stack);   // 스택의 전체 데이터 출력
```

스택 구현

백문이불여일타(百聞而不如一打)

- 스택의 이해
- 스택 응용
- **스택 구현**
  - 순차 자료구조
  - 연결 자료구조



스택 구현(Python): 순차 자료 구조 (1/7)

- **스택 구현:** 순차 자료구조

```
# 클래스 설계: ArrayStack
class ArrayStack:
    def __init__(self):          # 빈 스택 생성
        self.__stack = []

    def __del__(self):           # 스택 삭제

    def empty(self) -> bool:                    # 빈 스택 여부
    def size(self) -> int:                      # 스택의 원소 개수
    def push(self, num) -> None:                # 데이터 삽입
    def pop(self) -> None:                      # 데이터 삭제
    def top(self):                  # 스택에서 맨 위의 데이터 반환
    def printStack(self) -> None:   # 스택의 전체 데이터 출력
```

스택 구현(Python): 순차 자료 구조 (2/7)

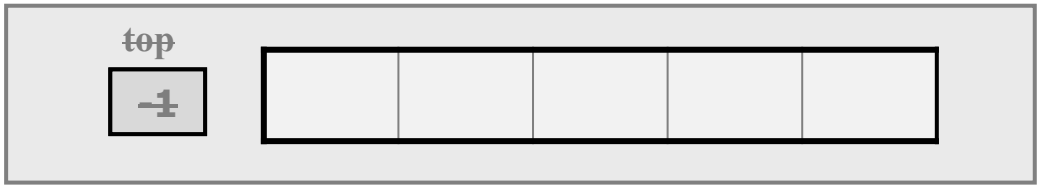
- **스택 구현:** 순차 자료구조
  - 프로그램 실행 결과는 다음과 같다.

```
 ### 스택 구현: 1차원 배열 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 3

      ##### 입력된 데이터 #####
STACK [ 5 4 3 2 1 ]
```

```
*IDLE Shell 3.11.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>>
================ RESTART: C:WUsersWclickW

 ### 스택 구현: 1차원 배열 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 1
삽입 할 데이터 입력(종료: 0): 1
삽입 할 데이터 입력(종료: 0): 2
삽입 할 데이터 입력(종료: 0): 3
삽입 할 데이터 입력(종료: 0): 4
삽입 할 데이터 입력(종료: 0): 5
삽입 할 데이터 입력(종료: 0): 0
```

```
C:₩WINDOWS₩system3
계속하려면 아무 키나 누르십시오 . . .
```

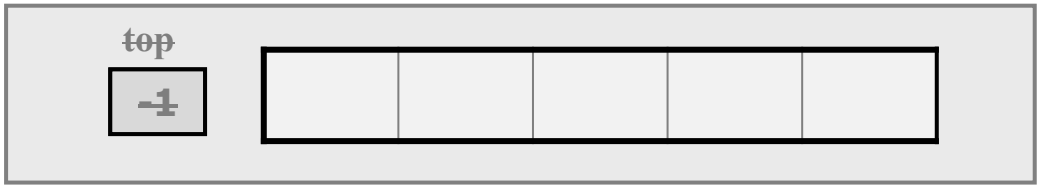
```
      ##### 입력된 데이터 #####
STACK [ 5 4 3 2 1 ]

 ### 스택 구현: 1차원 배열 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 2

삭제 된 데이터: 5
```



스택 구현(Python): 순차 자료 구조 (3/7)

예제 5-5: 스택 -- 순차 자료구조 ArrayStack.py (1/5)

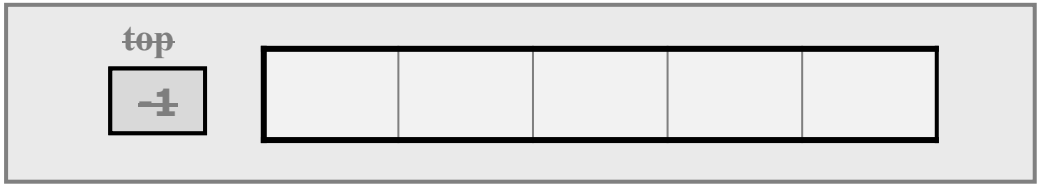
```
class ArrayStack:
    # 빈 스택 생성
    def __init__(self):
        self.__stack = []

    # 스택 삭제
    def __del__(self):
        self.__stack.clear()

    # 빈 스택 여부
    def empty(self) -> bool:
        if not self.__stack:
            return True
        return False

    # 스택의 원소 개수
    def size(self) -> int:
        return len(self.__stack)
```



스택 구현(Python): 순차 자료 구조 (4/7)

예제 5-5: 스택 -- 순차 자료구조 ArrayStack.py (2/5)

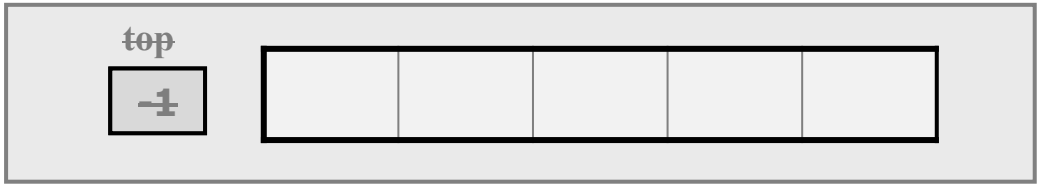
```
# 데이터 삽입: 스택에 새로운 데이터 추가
def push(self, num) -> None:
    self.__stack.append(num)

# 데이터 삭제: 스택에서 맨 위의 데이터 삭제
def pop(self) -> None:
    self.__stack.pop()

# 스택에서 맨 위의 데이터 확인
def top(self):
    if not self.__stack:
        return None
    return self.__stack[-1]
```



스택 구현(Python): 순차 자료 구조 (5/7)

예제 5-5: 스택 -- 순차 자료구조 ArrayStack.py (3/5)

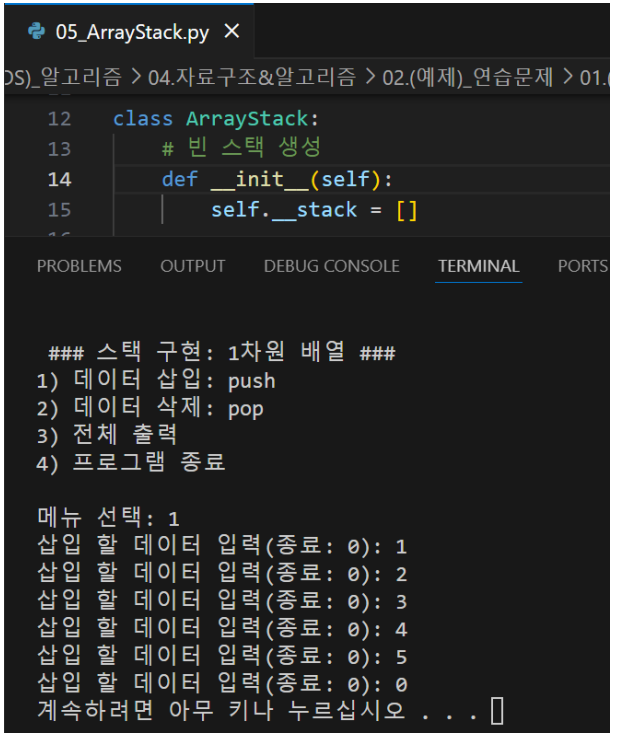
```
# 스택의 전체 데이터  출력
def printStack(self) -> None:
    if not self.__stack:
        print('입력된 데이터가 없습니다!!!')

    print('\n\t##### 입력된 데이터 #####')
    print('STACK [', end = ' ')
    for i in range(len(self.__stack) - 1, -1, -1):
        print(self.__stack[i], end = ' ')
    print(']')
```



스택 구현(Python): 순차 자료 구조 (6/7)

예제 5-5: 스택 -- 순차 자료구조 ArrayStack.py (4/5)

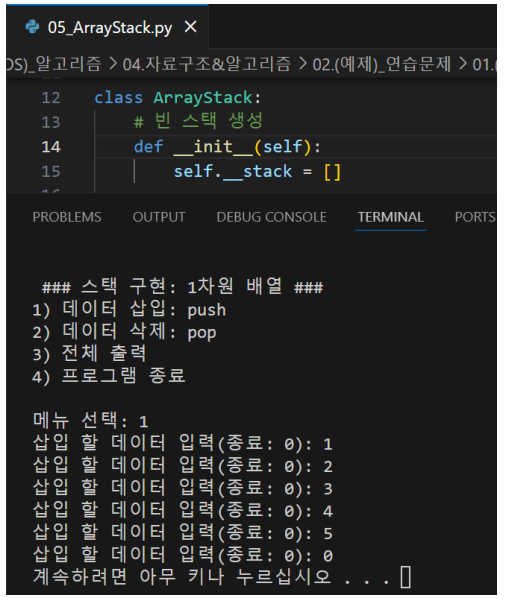
```
if __name__ == '__main__':
    import os          # system
    import sys         # exit

    s = ArrayStack()
    while True:
        os.system('cls')
        print('\n ### 스택 구현: 1차원 배열 ###')
        print('1) 데이터 삽입: push')
        print('2) 데이터 삭제: pop')
        print('3) 전체 출력')
        print('4) 프로그램 종료\n')
        print('메뉴 선택: ', end='')
        choice = int(input())
```



스택 구현(Python): 순차 자료 구조 (7/7)

예제 5-5: 스택 -- 순차 자료구조 ArrayStack.py (5/5)

```
        match choice:
            case 1:
                while True:
                    num = int(input('삽입 할 데이터 입력(종료: 0): '))
                    if num == 0:
                        break
                    s.push(num)
            case 2:
                print(f'\n삭제 된 데이터: {s.top()} ')
                s.pop()
            case 3:
                s.printStack()
            case 4:
                sys.exit("₩n프로그램 종료!!!")
            case _: print('\n잘못 선택 하셨습니다. \n')
        os.system('pause')

    # del s
    # s.__del__
```



스택 구현(C++): 순차 자료 구조 (1/9)

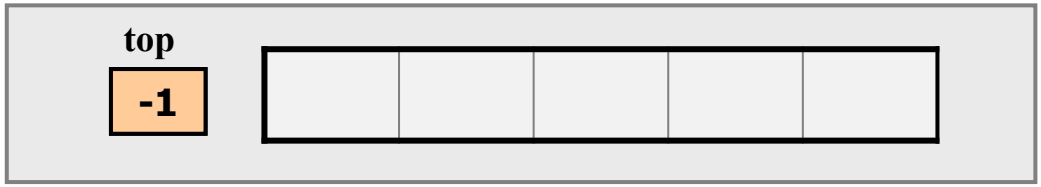
- **스택 구현:** 순차 자료구조

```
// #pragma once
#ifndef __ArrayStack_H__
#define __ArrayStack_H__

// 클래스 설계: ArrayStack
template <typename T>
class ArrayStack {
public:
        ArrayStack(int size = 10);              // 빈 스택 생성
        ~ArrayStack(void);                      // 스택 삭제
        bool    empty(void) const;              // 빈 스택 여부
        bool    full(void) const;               // 포화 상태 여부
        int     size(void) const;               // 원소 개수
        void    push(const T &data);            // 데이터 삽입
        void    pop(void);                      // 데이터 삭제
        T       top(void) const;                // 맨 위의 데이터 반환
        void    printStack(void) const;         // 전체 데이터 출력
private:
        int     top_;
        int     maxSize_;
        T       *stack_;
};

#endif
```

스택 구현(C++): 순차 자료 구조 (2/9)

- **스택 구현:** 순차 자료구조
  - 프로그램 실행 결과는 다음과 같다.

```
C:₩Users₩click₩OneDrive₩Cl

        ### 스택 구현: 1차원 배열 ###

1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 데이터 전체 출력
4) 프로그램 종료

메뉴 선택: 1
데이터 입력(종료: 0): 1
데이터 입력(종료: 0): 2
데이터 입력(종료: 0): 3
데이터 입력(종료: 0): 4
데이터 입력(종료: 0): 5
데이터 입력(종료: 0): 0
계속하려면 아무 키나 누르십시오 . . .
```




```
C:₩Users₩click₩OneDrive₩Cl

        ### 스택 구현: 1차원 배열 ###

1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 데이터 전체 출력
4) 프로그램 종료

메뉴 선택: 3

        ##### 입력된 데이터 #####

STACK [  5  4  3  2  1 ]
계속하려면 아무 키나 누르십시오 . . .
```

```
C:₩Users₩click₩OneDrive₩Cl

        ### 스택 구현: 1차원 배열 ###

1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 데이터 전체 출력
4) 프로그램 종료

메뉴 선택: 2
삭제 된 데이터:   5
계속하려면 아무 키나 누르십시오 . . .
```

스택 구현(C++): 순차 자료 구조 (3/9)

예제 5-6: 스택 -- 순차 자료구조 ArrayStack(demo).cpp (1/2)

```
#include <iostream>
#include "ArrayStack.cpp"                    // ArrayStack
// #include <conio.h>                        // _getch, _getche
using namespace std;

int main(void)
{
        int                     num, choice;
        ArrayStack<int>         s = ArrayStack<int>();

        while (true) {
                system("cls");
                cout << "\n\t### 스택 구현: 1차원 배열 ### \n\n" << endl;
                cout << "1) 데이터 삽입: push" << endl;
                cout << "2) 데이터 삭제: pop" << endl;
                cout << "3) 전체 출력" << endl;
                cout << "4) 프로그램 종료 \n" << endl;
                cout << "메뉴 선택: ";
                cin >> choice;
```

스택 구현(C++): 순차 자료 구조 (4/9)

예제 5-6: 스택 -- 순차 자료구조 ArrayStack(demo).cpp (2/2)

```
            switch (choice) {
                case 1:
                    while (true) {
                        cout << "데이터 입력(종료: 0): ";
                        cin >> num;
                        if (num == 0)
                            break;
                        s.push(num);
                    }
                    break;
                case 2: cout << "삭제 된 데이터: " << s.top() << endl;
                    s.pop();
                    break;
                case 3: s.printStack();
                    break;
                case 4: cout << "프로그램 종료!!!" << endl;
                    exit(0);        // return 0;
                default: cout << "잘못 선택 하셨습니다!!!" << endl;
            }
            // print("계속하려면 아무 키나 누르십시오...");
            // _getch();
            system("pause");
        }
        // s.~ArrayStack();
        return 0;
}
```

스택 구현(C++): 순차 자료 구조 (5/9)

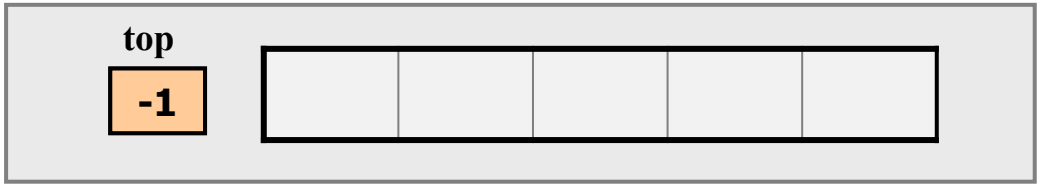
예제 5-6: 스택 -- 순차 자료구조 ArrayStack.cpp (1/5)

```
#include <iostream>
using namespace std;

// #pragma once
#ifndef __ArrayStack_H__
#define __ArrayStack_H__

// 클래스 설계: ArrayStack
template <typename T>
class ArrayStack {
public:
    ArrayStack(int size = 10);          // 빈 스택 생성
    ~ArrayStack(void);                  // 스택 삭제
    bool  empty(void) const;            // 빈 스택 여부
    bool  full(void) const;             // 포화 상태 여부
    int   size(void) const;             // 원소 개수
    void  push(const T &data);          // 데이터 삽입
    void  pop(void);                    // 데이터 삭제
    T     top(void) const;              // 맨 위의 데이터 반환
    void  printStack(void) const;       // 전체 데이터 출력
private:
    int   top_;
    int   maxSize_;
    T     *stack_;
};
```

스택 구현(C++): 순차 자료 구조 (6/9)

예제 5-6: 스택 -- 순차 자료구조 ArrayStack.cpp (2/5)

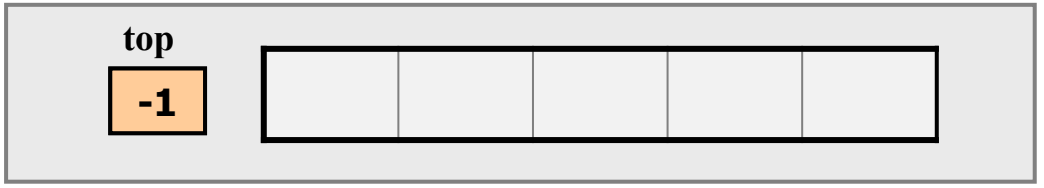
```
// 빈 스택 생성
template <typename T>
ArrayStack<T>::ArrayStack(int size) : top_(-1), maxSize_(size) {
        stack_ = new T[maxSize_];
}

// 스택 삭제
template <typename T>
ArrayStack<T>::~ArrayStack(void) {
        delete[] stack_;
}
```



스택 구현(C++): 순차 자료 구조 (7/9)

예제 5-6: 스택 -- 순차 자료구조 ArrayStack.cpp (3/5)

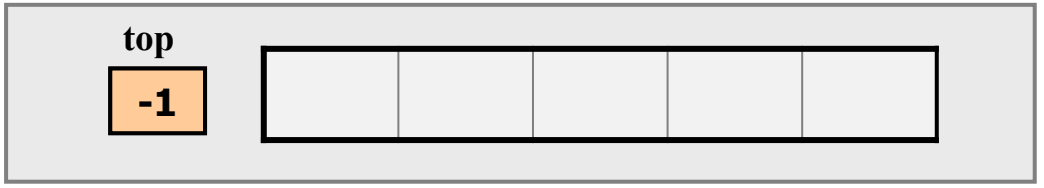
```
// 빈 스택 여부
template <typename T>
bool      ArrayStack<T>::empty(void) const {
          return top_ == -1;
}

// 포화 상태 여부
template <typename T>
bool      ArrayStack<T>::full(void) const {
          return top_ == maxSize_ - 1;
}

// 원소 개수
template <typename T>
int       ArrayStack<T>::size(void) const {
          return top_ + 1;
}
```



스택 구현(C++): 순차 자료 구조 (8/9)

예제 5-6: 스택 -- 순차 자료구조 ArrayStack.cpp (4/5)

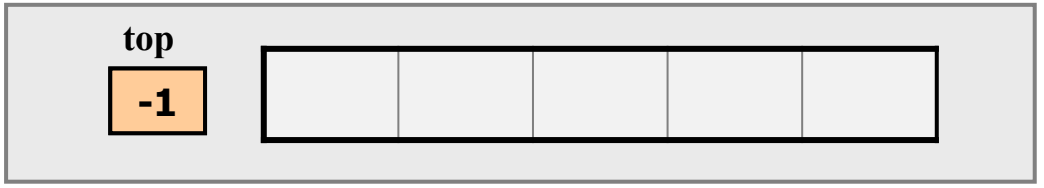
```
// 데이터 삽입: 스택에 새로운 데이터 추가
template <typename T>
void    ArrayStack<T>::push(const T &data) {
        if (full()) {
                return;
        }
        stack_[++top_] = data;
}

// 데이터 삭제: 스택에서 맨 위의 데이터 삭제
template <typename T>
void    ArrayStack<T>::pop(void) {
        if (empty()) {
                return;
        }
        --top_;
}

// 맨 위의 데이터 반환
template <typename T>
T       ArrayStack<T>::top(void) const {
        if (empty()) {
                return T();
        }
        return stack_[top_];
}
```



스택 구현(C++): 순차 자료 구조 (9/9)

예제 5-6: 스택 -- 순차 자료구조 ArrayStack.cpp (5/5)

```
// 전체 데이터 출력
template <typename T>
void    ArrayStack<T>::printStack(void) const {
        if (empty()) {
                cout << "입력된 데이터가 없습니다!!!" << endl;
                return;
        }

        cout << "\n\t##### 입력된 데이터 #####\n" << endl;
        cout << "STACK [";
        for (int i = top_; i >= 0; --i) {
                cout.width(3);
                cout << stack_[i];
        }
        cout << " ]" << endl;
}

#endif
```



스택 구현(C): 순차 자료 구조 (1/10)

- **스택 구현:** 순차 자료구조

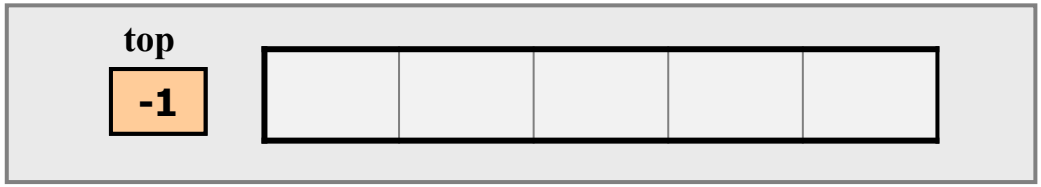
```
// #pragma once
#define                 StackMAXSIZE 1024
typedef     int         element;

#ifndef __ArrayStack_H__
#define __ArrayStack_H__

// 구조체 설계: ArrayStack
typedef struct _ArrayStack {
    int                 top;
    element             stack[StackMAXSIZE];
} ArrayStack;

#endif

// ArrayStack: 스택 생성 및 조작 함수
ArrayStack  *stackCreate(void);                          // 빈 스택 생성
void        stackDestroy(ArrayStack *Stack);             // 스택 삭제
_Bool       stackEmpty(ArrayStack *Stack);               // 빈 스택 여부
_Bool       stackFull(ArrayStack *Stack);                // 스택의 포화 상태 여부
Int         stackSize(ArrayStack *Stack);                // 스택의 원소 개수
Void        push(ArrayStack *Stack, element data);       // 데이터 삽입
Void        pop(ArrayStack *Stack);                      // 데이터 삭제
Element     top(ArrayStack *Stack);          // 스택에서 맨 위의 데이터 확인
Void        printStack(ArrayStack *Stack);   // 스택의 모든 데이터 출력
```

스택 구현(C): 순차 자료 구조 (2/10)

- **스택 구현:** 순차 자료구조
  - 프로그램 실행 결과는 다음과 같다.

```
C:₩Users₩click₩OneDrive₩Cl

        ### 스택 구현: 1차원 배열 ###

1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 데이터 전체 출력
4) 프로그램 종료

메뉴 선택: 1
데이터 입력(종료: 0): 1
데이터 입력(종료: 0): 2
데이터 입력(종료: 0): 3
데이터 입력(종료: 0): 4
데이터 입력(종료: 0): 5
데이터 입력(종료: 0): 0
계속하려면 아무 키나 누르십시오 . . .
```

```
C:₩Users₩click₩OneDrive₩Cl

        ### 스택 구현: 1차원 배열 ###

1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 데이터 전체 출력
4) 프로그램 종료

메뉴 선택: 3

        ##### 입력된 데이터 #####

STACK [  5  4  3  2  1 ]
계속하려면 아무 키나 누르십시오 . . .
```




```
C:₩Users₩click₩OneDrive₩Cl

        ### 스택 구현: 1차원 배열 ###

1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 데이터 전체 출력
4) 프로그램 종료

메뉴 선택: 2
삭제 된 데이터:   5
계속하려면 아무 키나 누르십시오 . . .
```

스택 구현(C): 순차 자료 구조 (3/10)

예제 5-7: 스택 -- 순차 자료구조 ArrayStack(demo).c (1/2)

```c
#include <stdio.h>
#include <stdbool.h>          // bool, true, false
#include <stdlib.h>           // system, exit
#include <conio.h>            // _getch, _getche
#include "ArrayStack.h"       // ArrayStack

int main(void)
{
        int     num, choice;

        // 빈 스택 생성: stack[StackMAXSIZE], top = -1
        ArrayStack      *s = stackCreate();

        while (true) {
                system("cls");
                printf("\n\t### 스택 구현: 1차원 배열 ### \n\n");
                printf("1) 데이터 삽입: push \n");
                printf("2) 데이터 삭제: pop \n");
                printf("3) 데이터 전체 출력 \n");
                printf("4) 프로그램 종료 \n\n");
                printf("메뉴 선택: ");
                scanf_s("%d%*c", &choice);
                // scanf("%d", &choice);
                // while (getchar() != EOF);
```

스택 구현(C): 순차 자료 구조 (4/10)

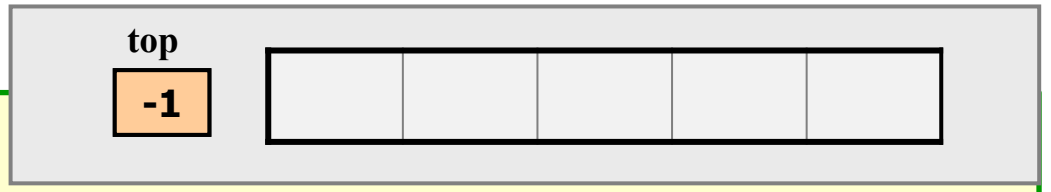
예제 5-7: 스택 -- 순차 자료구조 ArrayStack(demo).c (2/2)

```
            switch (choice) {
                    case 1:
                            while (true) {
                                    printf("데이터 입력(종료: 0): ");
                                    scanf_s("%d%*c", &num);
                                    // scanf("%d", &num);
                                    // while (getchar() != EOF);
                                    if (num == 0)
                                            break;
                                    push(s, num);
                            }
                            break;
                    case 2: printf("삭제 된 데이터: %3d \n", top(s) );
                            pop(s);
                            break;
                    case 3: printStack(s);
                            break;
                    case 4: printf("프로그램 종료!!!\n");
                            exit(0);        // return 0;
                    default: printf("잘못 선택 하셨습니다!!!\n");
            }
            // print("계속하려면 아무 키나 누르십시오...");
            // _getch();
            system("pause");
    }
    // stackDestroy(s);
    return 0;
}
```

스택 구현(C): 순차 자료 구조 (5/10)

예제 5-7: 스택 -- 순차 자료구조 ArrayStack.h

```
// #pragma once
#define                 StackMAXSIZE 1024
typedef         int     element;

#ifndef __ArrayStack_H__
#define __ArrayStack_H__

// 구조체 설계: ArrayStack
typedef struct _ArrayStack {
        int             top;
        element         stack[StackMAXSIZE];
} ArrayStack;

#endif
```

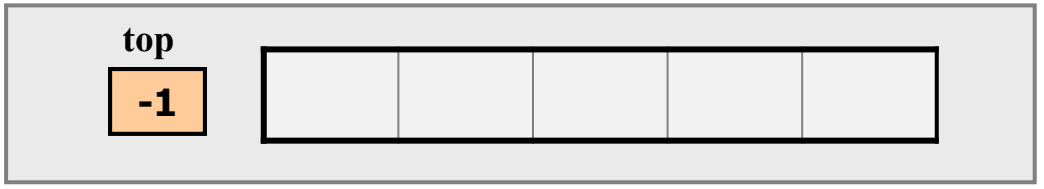



```
// ArrayStack: 스택 생성 및 조작 함수
ArrayStack      *stackCreate(void);                     // 빈 스택 생성
Void            stackDestroy(ArrayStack *Stack);        // 스택 삭제
_Bool           stackEmpty(ArrayStack *Stack);          // 빈 스택 여부
_Bool           stackFull(ArrayStack *Stack);           // 스택의 포화 상태 여부
Int             stackSize(ArrayStack *Stack);                   // 스택의 원소 개수
Void            push(ArrayStack *Stack, element data);          // 데이터 삽입
Void            pop(ArrayStack *Stack);                         // 데이터 삭제
element         top(ArrayStack *Stack);                 // 스택에서 맨 위의 데이터 반환
void            printStack(ArrayStack *Stack);          // 스택의 모든 데이터 출력
```

스택 구현(C): 순차 자료 구조 (6/10)

예제 5-7: 스택 -- 순차 자료구조 ArrayStack.c (1/5)

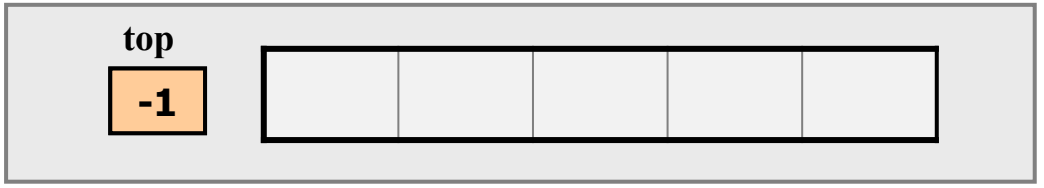
```
#include <stdio.h>
#include <stdbool.h>          // bool, true, false
#include <stdlib.h>           // malloc, free
#include "ArrayStack.h"       // ArrayStack

// 빈 스택 생성
ArrayStack      *stackCreate(void) {
        ArrayStack      *Stack = (ArrayStack *)malloc(sizeof(ArrayStack));
        if (Stack == NULL) {
                printf("스택 생성 실패!!! \n");
                return NULL;
        }
        Stack->top = -1;
        return Stack;
}

// 스택 삭제
void    stackDestroy(ArrayStack *Stack) {
        free(Stack);
}
```



스택 구현(C): 순차 자료 구조 (7/10)

예제 5-7: 스택 -- 순차 자료구조 ArrayStack.c (2/5)

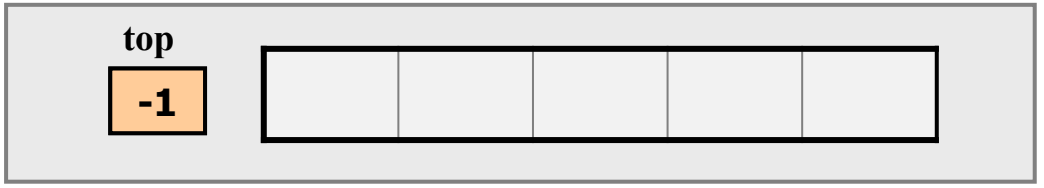
```
// 빈 스택 여부
_Bool    stackEmpty(ArrayStack *Stack) {
         if (Stack->top == -1)
                  return true;
         return false;
}

// 스택의 포화 상태 여부
_Bool    stackFull(ArrayStack *Stack) {
         if (Stack->top + 1 == StackMAXSIZE)
                  return true;
         return false;
}
```



스택 구현(C): 순차 자료 구조 (8/10)

예제 5-7: 스택 -- 순차 자료구조 ArrayStack.c (3/5)

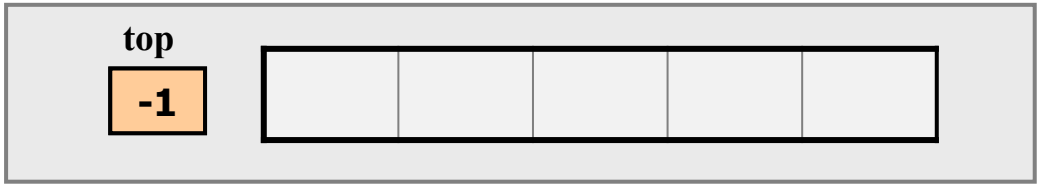
```
// 스택의 원소 개수
int     stackSize(ArrayStack *Stack) {
        return Stack->top + 1;
}

// 데이터 삽입: 스택에 새로운 데이터 추가
void    push(ArrayStack *Stack, element data) {
        // if (stackFull(Stack))     return;
        if (Stack->top + 1 == StackMAXSIZE)
                return;

        Stack->stack[++Stack->top] = data;
}
```



스택 구현(C): 순차 자료 구조 (9/10)

예제 5-7: 스택 -- 순차 자료구조 ArrayStack.c (4/5)

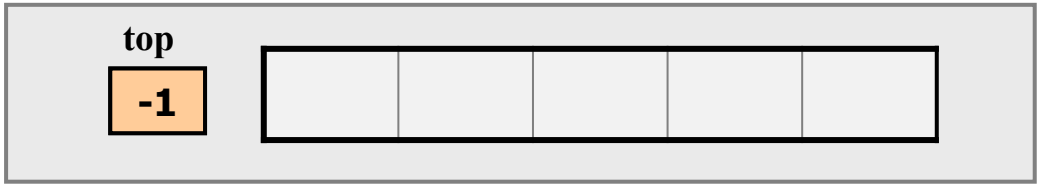
```
// 데이터 삭제: 스택에서 맨 위의 데이터 삭제
Void    pop(ArrayStack *Stack) {
        // if (stackEmpty(Stack))    return;
        if (Stack->top == -1)
                return;

        --Stack->top;
}

// 스택에서 맨 위의 데이터 반환
element top(ArrayStack *Stack) {
        // if (stackEmpty(Stack))    return -1;
        if (Stack->top == -1)
                return -1;

        return Stack->stack[Stack->top];
}
```



스택 구현(C): 순차 자료 구조 (10/10)

예제 5-7: 스택 -- 순차 자료구조 ArrayStack.c (5/5)

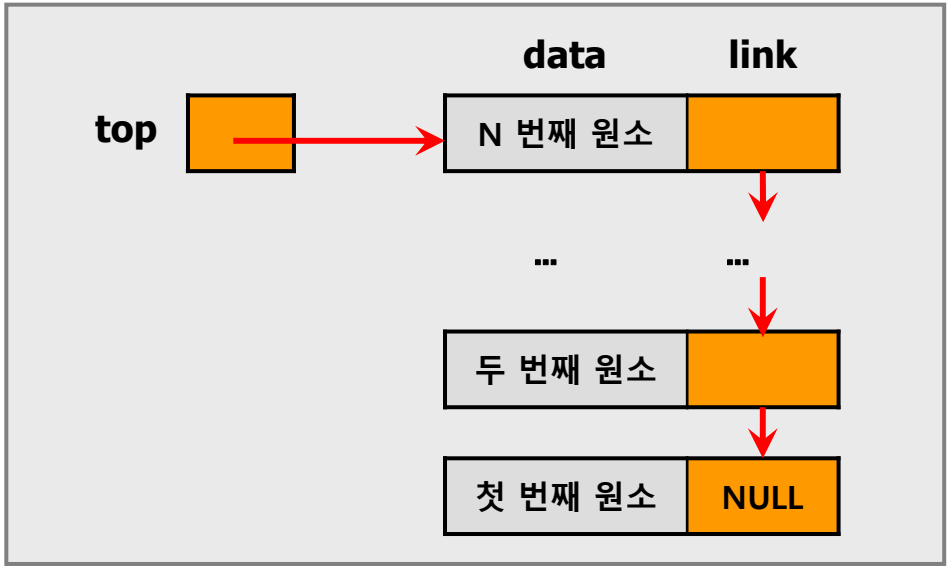
```
// 스택의 전체 데이터 출력
void    printStack(ArrayStack *Stack) {
        // if (stackEmpty(Stack)) {
        if (Stack->top == -1) {
                printf("입력된 데이터가 없습니다!!!\n");
                return;
        }

        printf("\n\t##### 입력된 데이터 #####\n\n");
        printf("STACK [");
        for (int i = Stack->top; i >=0; --i)
                printf("%3d", Stack->stack[i]);
        printf(" ]\n");
}
```



스택 구현

백문이불여일타(百聞而不如一打)

- 스택의 이해
- 스택 응용
- **스택 구현**
  - 순차 자료구조
  - **연결 자료구조**



스택 구현(Python): 연결 자료 구조 (1/5)

- **스택 구현:** 연결 자료구조

```
# 클래스 설계: LinkedStack
class LinkedStack:
    class SNode:
        def __init__(self, data, link=None):
            self.data = data
            self.link = link

    # 빈 스택 생성
    def __init__(self):
        self.__top = None
        self.__count = 0

    def __del__(self):                      # 스택 삭제
    def empty(self) -> bool:                # 빈 스택 여부
    def size(self) -> int:                  # 스택의 원소 개수
    def push(self, data) -> None:           # 데이터 삽입
    def pop(self) -> None:                  # 데이터 삭제
    def top(self):                          # 맨 위의 데이터 반환
    def printStack(self) -> None:           # 스택의 전체 데이터 출력
```

스택 구현(Python): 연결 자료 구조(2/5)

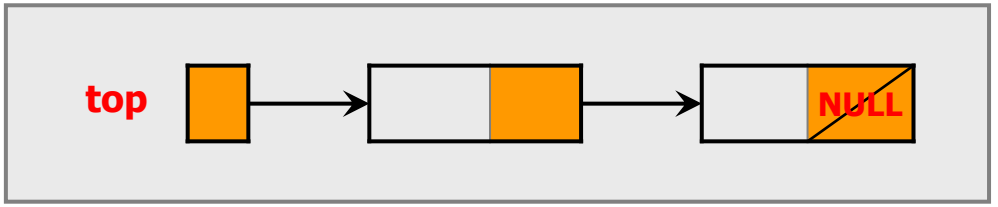
- **스택 구현:** 연결 자료구조
  - 프로그램 실행 결과는 다음과 같다.

```
 ### 스택 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 3

        ##### 입력된 데이터 #####

 STACK [ 5 4 3 2 1 ]
```

```
 ### 스택 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 2

삭제 된 데이터: 5
```

```
*IDLE Shell 3.11.2*
File  Edit  Shell  Debug  Options  Window  Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>>
================ RESTART: C:WUsersWclickW

 ### 스택 구현: 단순연결리스트 ###
1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 1
삽입 할 데이터 입력(종료: 0): 1
삽입 할 데이터 입력(종료: 0): 2
삽입 할 데이터 입력(종료: 0): 3
삽입 할 데이터 입력(종료: 0): 4
삽입 할 데이터 입력(종료: 0): 5
삽입 할 데이터 입력(종료: 0): 0
```

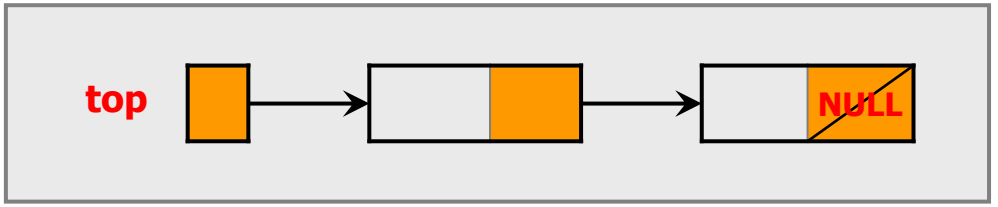
```
C:₩WINDOWS₩system3
계속하려면 아무 키나 누르십시오 . . .
```



스택 구현(Python): 연결 자료 구조 (3/5)

예제 5-8: 스택 -- 연결 자료구조 LinkedStack.py (1/3)

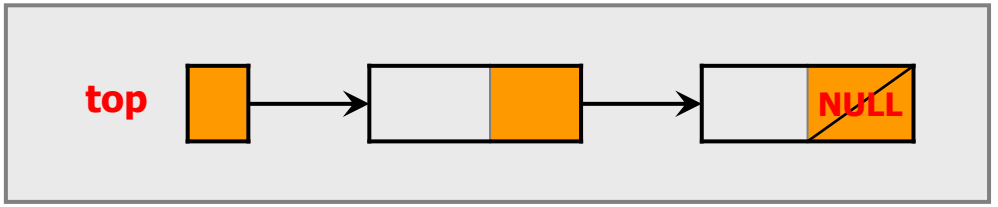
```
# 클래스 설계: LinkedStack
class LinkedStack:
    class SNode:
        def __init__(self, data, link=None):
            self.data = data
            self.link = link

    # 빈 스택 생성
    def __init__(self):
        self.__top = None
        self.__count = 0

    def __del__(self):                  # 스택 삭제: 모든 노드 삭제
    def empty(self) -> bool:            # 빈 스택 여부
    def size(self) -> int:              # 스택의 원소 개수
    def push(self, data) -> None:       # 데이터 삽입
    def pop(self) -> None:              # 데이터 삭제
    def top(self):                      # 맨 위의 데이터 반환
    def printStack(self) -> None:       # 스택의 전체 데이터 출력
```



스택 구현(Python): 연결 자료 구조 (4/5)

예제 5-8: 스택 -- 연결 자료구조 LinkedStack.py (2/3)

```
if __name__ == '__main__':
    import os           # system
    import sys          # exit

    # "SNode" is not defined
    # tNode = SNode(10, None)
    s = LinkedStack()
    while True:
        os.system('cls')
        print('\n ### 스택 구현: 단순연결리스트 ###')
        print('1) 데이터 삽입: push')
        print('2) 데이터 삭제: pop')
        print('3) 전체 출력')
        print('4) 프로그램 종료\n')
        print('메뉴 선택: ', end='')
        choice = int(input())
```

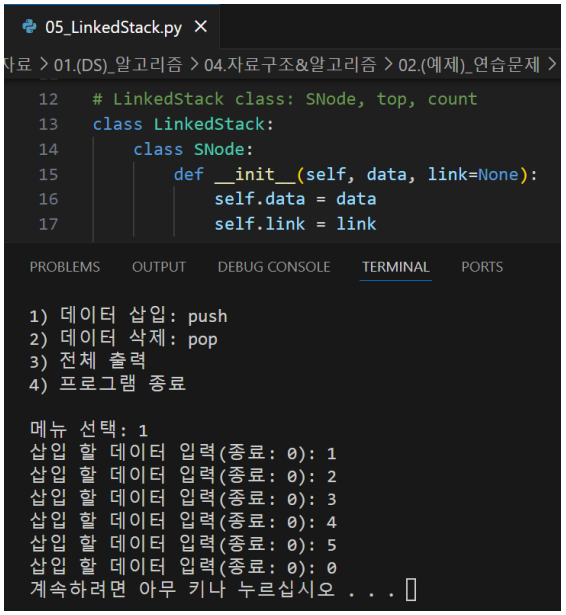



```
05_LinkedStack.py
자료 > 01.(DS)_알고리즘 > 04.자료구조&알고리즘 > 02.(예제)_연습문제 >
12  # LinkedStack class: SNode, top, count
13  class LinkedStack:
14      class SNode:
15          def __init__(self, data, link=None):
16              self.data = data
17              self.link = link

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 전체 출력
4) 프로그램 종료

메뉴 선택: 1
삽입 할 데이터 입력(종료: 0): 1
삽입 할 데이터 입력(종료: 0): 2
삽입 할 데이터 입력(종료: 0): 3
삽입 할 데이터 입력(종료: 0): 4
삽입 할 데이터 입력(종료: 0): 5
삽입 할 데이터 입력(종료: 0): 0
계속하려면 아무 키나 누르십시오 . . .
```

스택 구현(Python): 연결 자료 구조 (5/5)

예제 5-8: 스택 -- 연결 자료구조 LinkedStack.py (3/3)

```
        match choice:
            case 1:
                while True:
                    num = int(input('삽입 할 데이터 입력(종료: 0): '))
                    if num == 0:
                        break
                    s.push(num)
            case 2:
                print(f'\n삭제 된 데이터: {s.top()}')
                s.pop()
            case 3:
                s.printStack()
            case 4:
                sys.exit("₩n프로그램 종료!!!")
            case _: print('\n잘못 선택 하셨습니다. \n')
        os.system('pause')

    # del s
    # s.__del__
```



스택 구현(C++): 연결 자료 구조 (1/7)

- **스택 구현:** 연결 자료구조

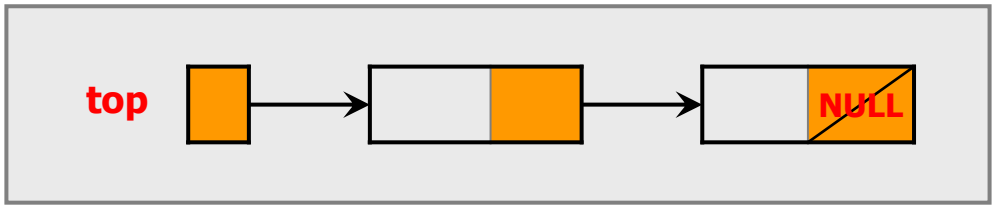
```
// LinkedQueue.cpp
#include "LinkedNode(template).cpp"

// #pragma once
#ifndef __LinkedStack_Template_H__
#define __LinkedStack_Template_H__

// 클래스 설계: LinkedStack
template <typename T>
class LinkedStack {
        LinkedStack();                          // 빈 스택 생성
        ~LinkedStack();                         // 스택 삭제
        bool    empty(void) const;              // 빈 스택 여부
        int     size(void) const;               // 원소 개수
        void    push(const T &data);            // 데이터 삽입
        void    pop(void);                      // 데이터 삭제
        T       top(void) const;                // 맨 위의 데이터 반환
        void    printStack(void) const;         // 전체 데이터 출력
private:
        SNode<T>            *top_;
        int                 count_;
};

#endif
```

```
// LinkedNode(template).cpp
// #pragma once
#ifndef __SNode_Template_H__
#define __SNode_Template_H__

template <typename T>    class LinkedQueue;

// 클래스 설계: SNode
template <typename T>
class SNode {
public:
        SNode(const T &data);
        T       getData(void) const;
private:
        T                   data_;
        SNode<T>            *link_;
        template <typename T> friend class LinkedQueue;
};

#endif
```

스택 구현(C++): 연결 자료 구조 (2/7)

- **스택 구현:** 연결 자료구조
  - 프로그램 실행 결과는 다음과 같다.

```
C:₩Users₩click₩OneDrive₩Cl

        ### 스택 구현: 단순연결리스트 ###

1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 데이터 전체 출력
4) 프로그램 종료

메뉴 선택: 1
데이터 입력(종료: 0): 1
데이터 입력(종료: 0): 2
데이터 입력(종료: 0): 3
데이터 입력(종료: 0): 4
데이터 입력(종료: 0): 5
데이터 입력(종료: 0): 0
계속하려면 아무 키나 누르십시오 . . .
```



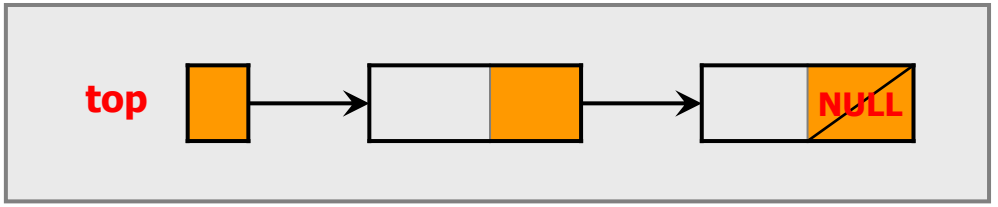

```
C:₩Users₩click₩OneDrive₩Cl

        ### 스택 구현: 단순연결리스트 ###

1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 데이터 전체 출력
4) 프로그램 종료

메뉴 선택: 3

        ##### 입력된 데이터 #####

STACK [  5  4  3  2  1 ]
계속하려면 아무 키나 누르십시오 . . .
```

```
C:₩Users₩click₩OneDrive₩Cl

        ### 스택 구현: 단순연결리스트 ###

1) 데이터 삽입: push
2) 데이터 삭제: pop
3) 데이터 전체 출력
4) 프로그램 종료

메뉴 선택: 2
삭제 된 데이터:   5
계속하려면 아무 키나 누르십시오 . . .
```

스택 구현(C++): 연결 자료 구조 (3/7)

예제 5-9: 스택 -- 연결 자료구조 LinkedStack(demo).cpp (1/2)

```
#include <iostream>
#include "LinkedStack.cpp"    // LinkedStack, SNode<T>
// #include <conio.h>         // _getch, _getche
using namespace std;

int main(void)
{
        int                   num, choice;
        LinkedStack<int>      s = LinkedStack<int>();

        while (true) {
                system("cls");
                cout << "\n\t### 스택 구현: 단순연결리스트 ### \n\n" << endl;
                cout << "1) 데이터 삽입: push" << endl;
                cout << "2) 데이터 삭제: pop" << endl;
                cout << "3) 전체 출력" << endl;
                cout << "4) 프로그램 종료 \n" << endl;
                cout << "메뉴 선택: ";
                cin >> choice;
```



스택 구현(C++): 연결 자료 구조 (4/7)

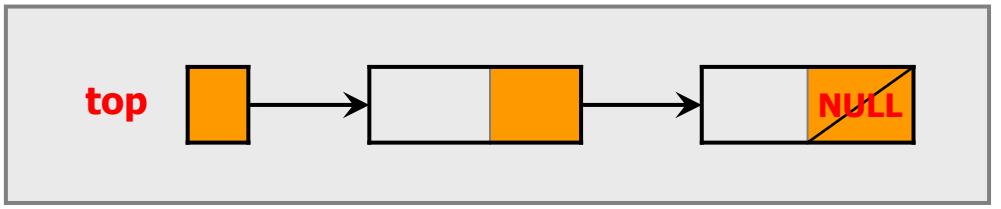
예제 5-9: 스택 -- 연결 자료구조 LinkedStack(demo).cpp (2/2)

```
                switch (choice) {
                        case 1:
                                while (true) {
                                        cout << "데이터 입력(종료: 0): ";
                                        cin >> num;
                                        if (num == 0)
                                                break;
                                        s.push(num);
                                }
                                break;
                        case 2: cout << "삭제 된 데이터: " << s.top() << endl;
                                s.pop();
                                break;
                        case 3: s.printStack();
                                break;
                        case 4: cout << "프로그램 종료!!!" << endl;
                                exit(0);        // return 0;
                        default: cout << "잘못 선택 하셨습니다!!!" << endl;
                }
                // print("계속하려면 아무 키나 누르십시오...");
                // _getch();
                system("pause");
        }
        // s.~LinkedStack();
        return 0;
}
```

스택 구현(C++): 연결 자료 구조 (5/7)

예제 5-9: 스택 -- 연결 자료구조 LinkedNode(template).cpp (1/2)

```
// #pragma once
#ifndef __SNode_Template_H__
#define __SNode_Template_H__
```

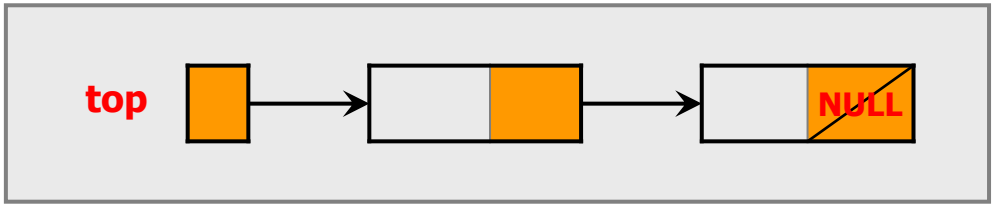



```
template <typename T> class LinkedStack;

// 클래스 설계: SNode
template <typename T>
class SNode {
public:
        SNode(const T &data);
        T       getData(void) const;
private:
        T               data_;
        SNode<T>        *link_;
        template <typename T> friend class LinkedStack;
};
```

스택 구현(C++): 연결 자료 구조 (6/7)

예제 5-9: 스택 -- 연결 자료구조 LinkedNode(template).cpp (2/2)

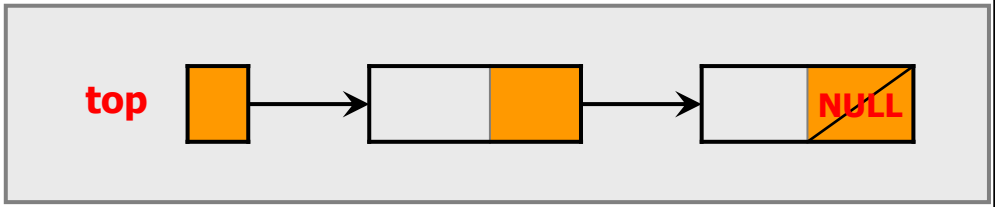
```
// SNode class: 노드 생성 및 조작 함수
template <typename T>
SNode<T>::SNode(const T &data) :
        data_(data), link_(nullptr) {}

template <typename T>
T  SNode<T>::getData(void) const {
        return data_;
}

#endif
```



스택 구현(C++): 연결 자료 구조 (7/7)

예제 5-9: 스택 -- 연결 자료구조 LinkedStack.cpp

```
#include <iostream>
#include "LinkedNode(template).cpp"          // SNode<T>
using namespace std;

// #pragma once
#ifndef __LinkedStack_Template_H__
#define __LinkedStack_Template_H__

// 클래스 설계: LinkedStack
template <typename T>
class LinkedStack {
public:
        LinkedStack();                       // 빈 스택 생성
        ~LinkedStack();                      // 스택 삭제
        bool    empty(void) const;           // 빈 스택 여부
        int     size(void) const;            // 원소 개수
        void    push(const T &data);         // 데이터 삽입
        void    pop(void);                   // 데이터 삭제
        T       top(void) const;             // 맨 위의 데이터 확인
        void    printStack(void) const;      // 전체 데이터 출력
private:
        SNode<T>        *top_;
        int             count_;
};

#endif
```



스택 구현(C): 연결 자료 구조 (1/9)

- **스택 구현:** 연결 자료구조

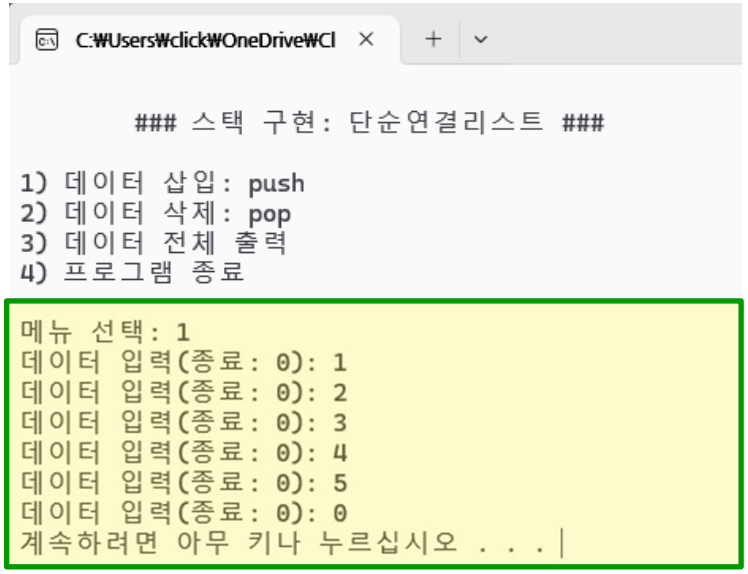
```
// #pragma once
#include "LinkedNode.h"                 // SNode

#ifndef __LinkedStack_H__
#define __LinkedStack_H__

// 구조체 설계: LinkedStack
typedef struct _LinkedStack {
        SNode       *top;
        int         count;
} LinkedStack;

#endif
```

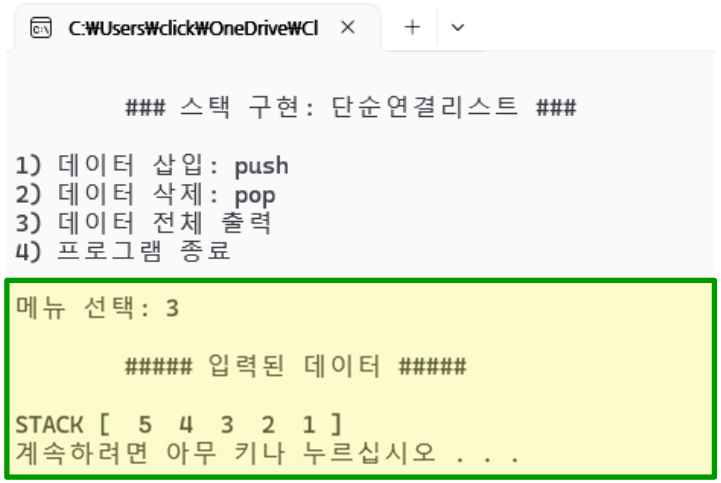

```
// 파일명: LinkedNode.h
// #pragma once
typedef     int         element;

#ifndef __SNode_H__
#define __SNode_H__

// 구조체 설계: SNode(data, link)
typedef struct _SNode {
        element         data;
        struct _SNode   *link;
} SNode;

#endif

SNode       *makeSNode(element data);
```

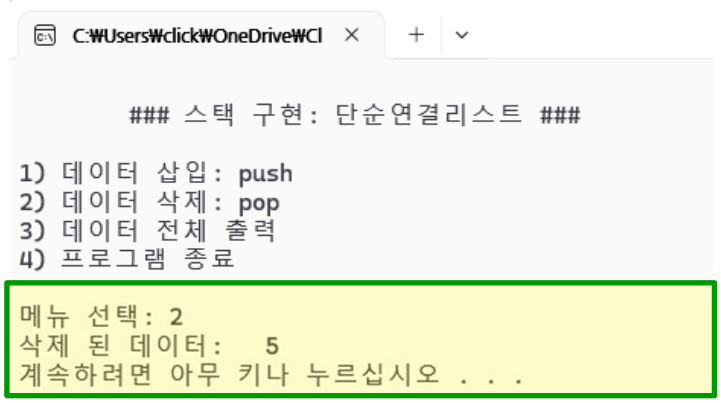
```
// LinkedStack: 스택 생성 및 조작 함수
LinkedStack *stackCreate(void);                     // 빈 스택 생성
void        stackDestroy(LinkedStack *Stack);       // 스택 삭제: 모든 노드 삭제
_Bool       stackEmpty(LinkedStack *Stack);         // 빈 스택 여부
int         stackSize(LinkedStack *Stack);          // 스택의 원소 개수
void        push(LinkedStack *Stack, element data); // 데이터 삽입
void        pop(LinkedStack *Stack);                // 데이터 삭제
element     top(LinkedStack *Stack);                // 스택에서 맨 위의 데이터 반환
void        printStack(LinkedStack *Stack);         // 스택의 전체 데이터 출력
```

스택 구현(C): 연결 자료 구조 (2/9)

- **스택 구현:** 연결 자료구조
  - 프로그램 실행 결과는 다음과 같다.







스택 구현(C): 연결 자료 구조 (3/9)

예제 5-10: 스택 -- 연결 자료구조 LinkedStack(demo).c (1/2)

```
#include <stdio.h>
#include <stdlib.h>                    // system
#include <stdbool.h>                   // bool, true, false
#include "LinkedStack.h"               // LinkedStack, SNode
// #include "LinkedNode.h"             // SNode

int main(void)
{
        int             num, choice;
        LinkedStack     *s = stackCreate();

        while (true) {
                system("cls");
                printf("\n\t### 스택 구현: 단순연결리스트 ### \n\n");
                printf("1) 데이터 삽입: push \n");
                printf("2) 데이터 삭제: pop \n");
                printf("3) 데이터 전체 출력 \n");
                printf("4) 프로그램 종료 \n\n");
                printf("메뉴 선택: ");
                scanf_s("%d%*c", &choice);   // scanf("%d", &choice);
                // while (getchar() != '\n');
```

스택 구현(C): 연결 자료 구조 (4/9)

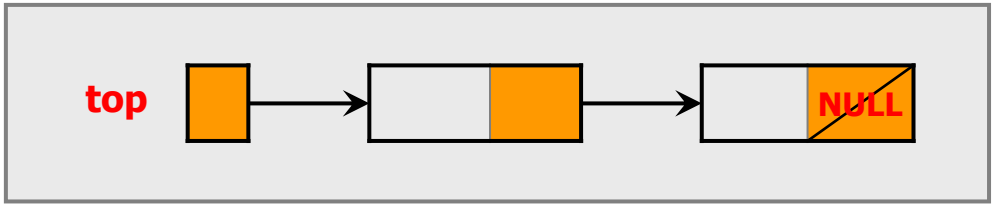
예제 5-10: 스택 -- 연결 자료구조 LinkedStack(demo).c (2/2)

```
            switch (choice) {
                    case 1:
                            while (true) {
                                    printf("데이터 입력(종료: 0): ");
                                    scanf_s("%d%*c", &num);
                                    // scanf("%d", &num);
                                    // while (getchar() != '\n');
                                    if (num == 0)
                                            break;
                                    push(s, num);
                            }
                            break;
                    case 2: printf("삭제 된 데이터: %3d \n", top(s) );
                            pop(s);
                            break;
                    case 3: printStack(s);
                            break;
                    case 4: printf("프로그램 종료!!!\n");
                            exit(0);        // return 0;
                    default: printf("잘못 선택 하셨습니다!!!\n");
            }
            // print("계속하려면 아무 키나 누르십시오...");
            // _getch();
            system("pause");
    }
    // stackDestroy(tStack);
    return 0;
}
```

스택 구현(C): 연결 자료 구조 (5/9)

예제 5-10: 스택 -- 연결 자료구조 LinkedStack.h

```
// #pragma once
#include "LinkedNode.h"       // SNode, makeSNode

#ifndef __LinkedStack_H__
#define __LinkedStack_H__

// 구조체 설계: LinkedStack
typedef struct _LinkedStack {
        SNode   *top;
        int     count;
} LinkedStack;

#endif
```

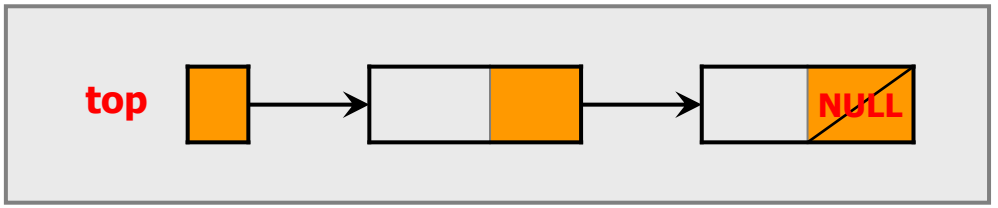



```
// LinkedStack: 스택 생성 및 조작 함수
LinkedStack     *stackCreate(void);                     // 빈 스택 생성
void            stackDestroy(LinkedStack *Stack);       // 스택 삭제: 모든 노드 삭제
_Bool           stackEmpty(LinkedStack *Stack);                 // 빈 스택 여부
int             stackSize(LinkedStack *Stack);                  // 스택의 원소 개수
Void            push(LinkedStack *Stack, element data);         // 데이터 삽입
Void            pop(LinkedStack *Stack);                        // 데이터 삭제
Element         top(LinkedStack *Stack);                // 스택에서 맨 위의 데이터 반환
void            printStack(LinkedStack *Stack);         // 스택의 전체 데이터 출력
```

스택 구현(C): 연결 자료 구조 (6/9)

예제 5-10: 스택 -- 연결 자료구조 LinkedStack.c (1/5)

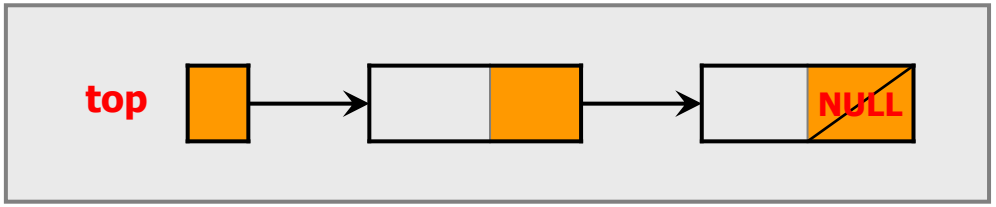
```
#include <stdio.h>
#include <stdlib.h>                    // malloc, free
#include "LinkedStack.h"               // LinkedStack, SNode
// #include "LinkedNode.h"             // SNode

// 빈 스택 생성
LinkedStack     *stackCreate(void) {
        LinkedStack     *Stack = (LinkedStack *)malloc(sizeof(LinkedStack));
        if (Stack == NULL) {
                printf("스택 생성 실패!!! \n");
                exit(100);
        }
        Stack->top = NULL;
        Stack->count = 0;

        return Stack;
}
```



스택 구현(C): 연결 자료 구조 (7/9)

예제 5-10: 스택 -- 연결 자료구조 LinkedStack.c (2/5)

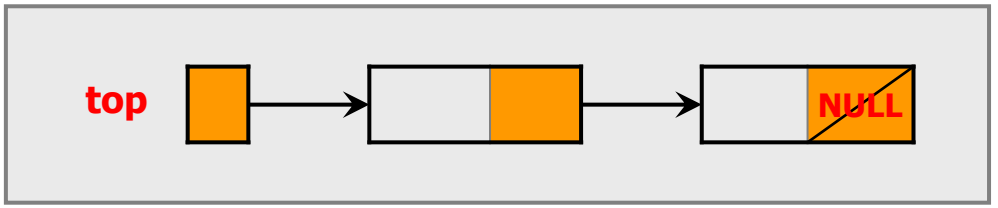
```
// 스택 삭제: 모든 노드 삭제
void    stackDestroy(LinkedStack *Stack) {
        SNode   *tNode = Stack->top;
        while (tNode) {
                Stack->top = tNode->link;
                free(tNode);
                tNode = Stack->top;
        }

        free(Stack);
        return;
}
```



스택 구현(C): 연결 자료 구조 (8/9)

예제 5-10: 스택 -- 연결 자료구조 LinkedStack.c (3/5)

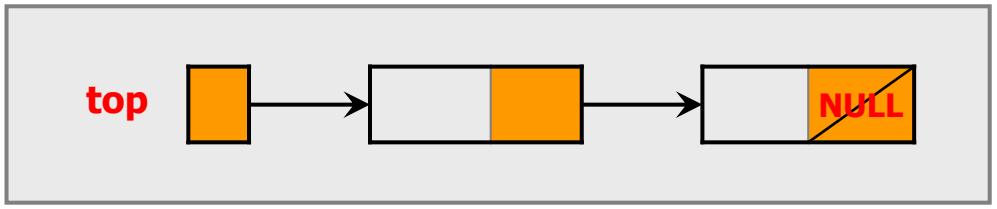
```
// 빈 스택 여부 확인
_Bool   stackEmpty(LinkedStack *Stack) {
        return Stack->top == NULL;
}

// 스택의 원소 개수
int     stackSize(LinkedStack *Stack) {
        return Stack->count;

}

// 데이터 삽입: 스택에 새로운 데이터 추가
void    push(LinkedStack *Stack, element data) {
        SNode   *newNode = makeSNode(data);

        newNode->link = Stack->top;
        Stack->top = newNode;

        ++Stack->count;
}
```



스택 구현(C): 연결 자료 구조 (9/10)

예제 5-10: 스택 -- 연결 자료구조 LinkedStack.c (4/5)

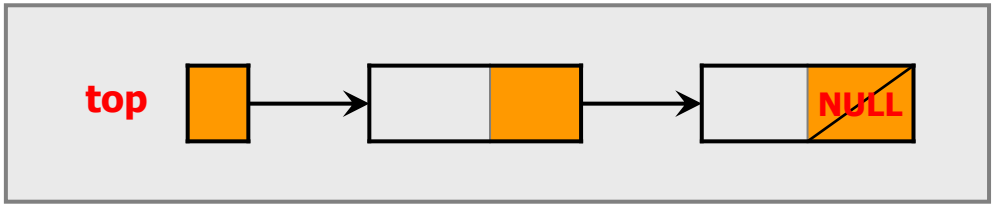
```
// 데이터 삭제: 스택에서 맨 위의 데이터 삭제
void    pop(LinkedStack *Stack) {
        // if (stackEmpty(Stack))
        if (Stack->top == NULL)
                return;

        SNode  *tNode = Stack->top;
        Stack->top = tNode->link;

        free(tNode);
        --Stack->count;
}

// 스택에서 맨 위의 데이터 반환
element top(LinkedStack *Stack) {
        // if (stackEmpty(Stack))
        if (Stack->top == NULL)
                return EOF;

        return Stack->top->data;
}
```



스택 구현(C): 연결 자료 구조 (10/10)

예제 5-10: 스택 -- 연결 자료구조 LinkedStack.c (5/5)

```
// 전체 데이터 출력
void    printStack(LinkedStack *Stack) {
        // if (stackEmpty(Stack)) {
        if (Stack->top == NULL) {
                printf("입력된 데이터가 없습니다!!!\n");
                return;
        }

        printf("\n\t##### 입력된 데이터 #####\n\n");
        printf("STACK [");

        SNode   *tNode = Stack->top;
        while (tNode) {
                printf("%3d", tNode->data);
                tNode = tNode->link;
        }
        printf(" ]\n");
}
```



참고문헌

[1] "이것이 자료구조+알고리즘이다: with C 언어", 박상현, 한빛미디어, 2022.

[2] "C++로 구현하는 자료구조와 알고리즘(2판)", Michael T. Goodrich, 김유성 외 2인 번역, 한빛아카데미, 2020.

[3] "IT CookBook, 쉽게 배우는 자료구조 with 파이썬", 문병로, 한빛아카데미, 2022.

[4] 문병로, "IT CookBook, 쉽게 배우는 알고리즘: 관계 중심의 사고법"(3판), 개정판, 한빛아카데미, 2024.

[5] "코딩 테스트를 위한 자료 구조와 알고리즘 with C++", John Carey 외 2인, 황선규 역, 길벗, 2020.

[6] "이것이 취업을 위한 코딩 테스트다 with 파이썬", 나동빈, 한빛미디어, 2020.

[7] "SW Expert Academy", SAMSUNG, 2025 of viewing the site, https://swexpertacademy.com/.

[8] "BAEKJOON", (BOJ) BaekJoon Online Judge, 2025 of viewing the site, https://www.acmicpc.net/.

[9] "programmers", grepp, 2025 of viewing the site, https://programmers.co.kr/.

[10] "goormlevel", goorm, 2025 of viewing the siteh, https://level.goorm.io/