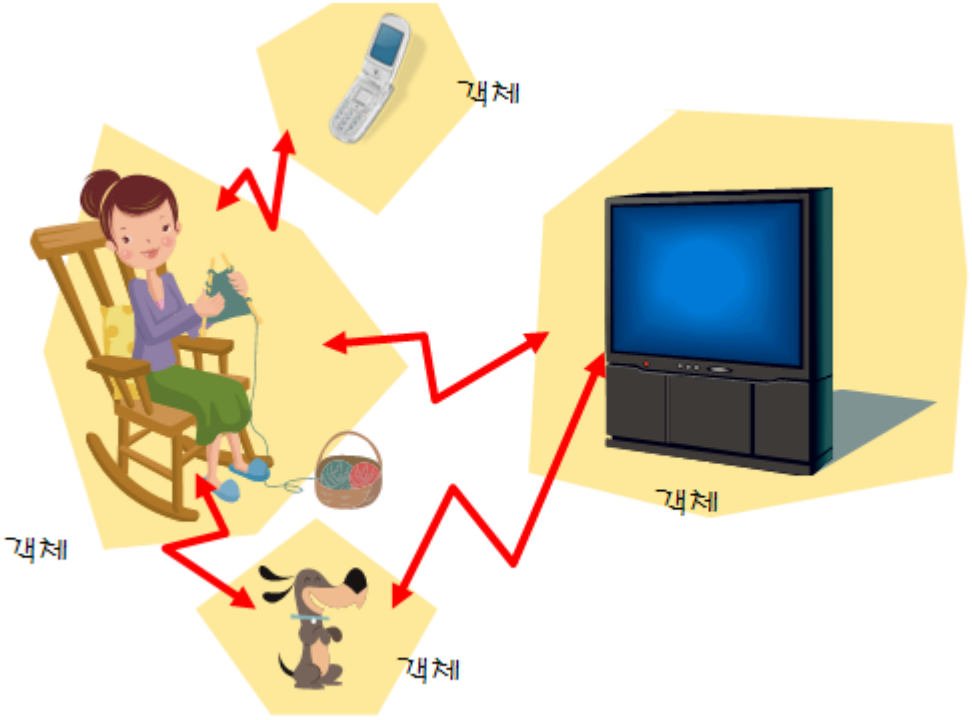
C++ Espresso

제12_2장 템플릿



이번 장에서 학습할 내용

- 클래스 템플릿
- 스택 예제

배열을 복사하는 제네릭 함수 mcopy() 함수 복습

두 개의 배열을 매개 변수로 받아 배열을 복사하는 제네릭 mcopy() 함수를 작성하라.

```
int main() {
    int x[] = { 1, 2, 3, 4, 5 }, y[5];
    char c[5] = { 'H', 'e', 'l', 'l', 'o' }, e[5];

    mcopy(x, y, 5); // int x[]의 원소 5개를 double d[]에 복사
    mcopy(c, e, 5);// char c[]의 원소 5개를 char e[]에 복사

    for (int i = 0; i<5; i++)
        cout << y[i] << ' ';
    cout << endl;

    for (int i = 0; i<5; i++)
        cout << e[i] << ' ';
    cout << endl;
}
```

```
1 2 3 4 5
H e l l o
```

정답

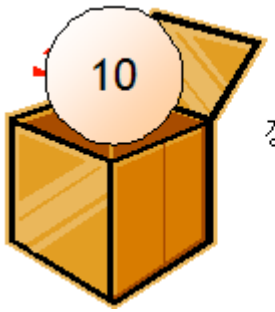
```
template <class T1, class T2>
void mcopy(T1 src[], T2 dest[], int n) {
    for (int i = 0; i<n; i++)
        dest[i] = (T2)src[i];
}
```

클래스 템플릿

- 클래스 템플릿(class template): 클래스를 찍어내는 틀(template)

```
template <typename 타입이름, ...>
class 클래스이름 {
...

}
```

- 예제: 하나의 값을 저장하고 있는 박스



정수를 저장하는 상자

예제

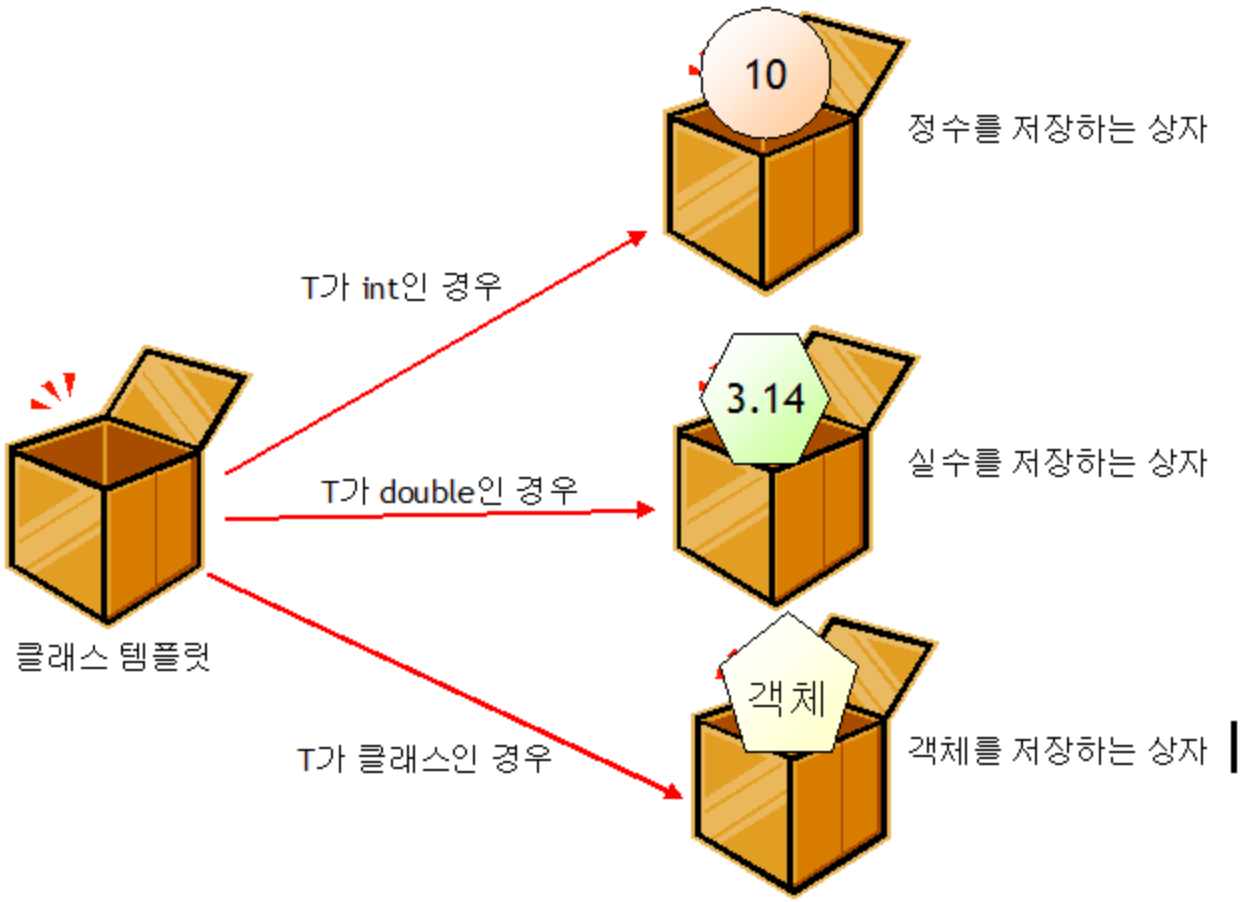
클래스

```
class Box{
    int data;
public:
    Box(){}
    void set(int value){
        data = value;
    }
    int get(){
        return data;
    }
};
```

메인함수

```
int main()
{
    Box box;
    box.set(100);
    cout << box.get() << endl;
    return 0;
}
```

클래스 템플릿 버전



예제

클래스

```
template <typename T>
class Box{
    T data;
public:
    Box(){}
    void set(T value){
        data = value;
    }
    T get(){
        return data;
    }
};
```

메인함수

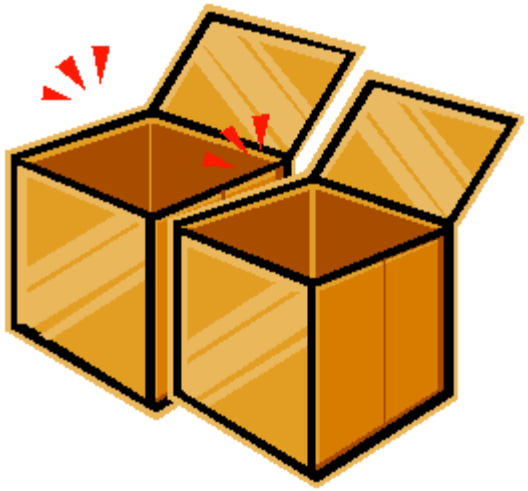
```
int main()
{
    Box<int> box;
    box.set(100);
    cout << box.get() << endl;

    Box<double> box1;
    box1.set(3.141592);
    cout << box1.get() << endl;
    return 0;
}
```

클래스 외부에 정의

```
template <typename T>
class Box{
    T data;
public:
    Box();
    void set(T value){
        data = value;
    }
    T get(){
        return data;
    }
};

template<typenamd T>
Box<T>::Box(){}
```

두개의 타입 매개 변수

- 두 개의 데이터를 저장하는 클래스 Box2



Box2 클래스 템플릿

실습

예제

```
10, 3.14
계속하려면 아무
```

```
template <typename T1, typename T2>
class Box2{
    T1 firstData;
    T2 secondData;

public:
    T1 getFirst(){ return firstData; }
    T2 getSecond(){ return secondData; }
    void setFirst(T1 value){ firstData = value; }
    void setSecond(T2 value){ secondData = value; }
};
```

```
int main()
{
    Box2<int, double> b;
    b.setFirst(10);
    b.setSecond(3.14);
    cout << b.getFirst() << ", " << b.getSecond() << endl;
    return 0;
}
```

템플릿 함수보다 중복 함수가 우선

```
template <class T>
void print(T array[], int n) {
    for (int i = 0; i<n; i++)
        cout << array[i] << '\t';
    cout << endl;
}
```

```
int main() {
    int x[] = { 65, 66, 67, 68, 69 };
    print(x, 5);

    double d[5] = { 65.1, 66.2, 67.3, 68.4, 69.5 };
    print(d, 5);

    char c[5] = { 65, 66, 67, 68, 69 };
    print(c, 5);
}
```

```
65      66      67      68      69
65.1    66.2    67.3    68.4    69.5
A       B       C       D       E
```

템플릿 함수보다 중복 함수가 우선

```
template <class T>
void print(T array[], int n) {
    for (int i = 0; i<n; i++)
        cout << array[i] << '\t';
    cout << endl;
}

void print(char array[], int n) {
    for (int i = 0; i<n; i++)
        cout << (int)array[i] << '\t';
    cout << endl;
}
```

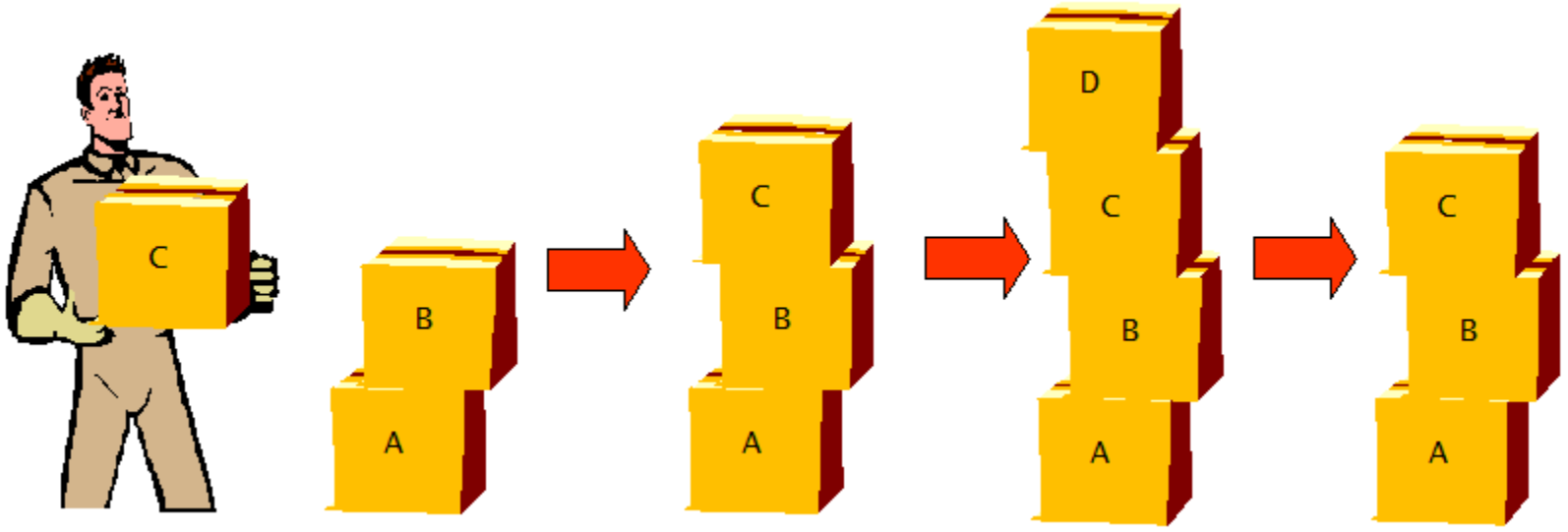
- 템플릿 함수와 이름이 동일한 함수가 중복되어 있을 때, **컴파일러는 중복된 함수를 템플릿 함수보다 우선하여 바인딩**한다.

```
int main() {
    int x[] = { 65, 66, 67, 68, 69 };
    print(x, 5);

    double d[5] = { 65.1, 66.2, 67.3, 68.4, 69.5 };
    print(d, 5);

    char c[5] = { 65, 66, 67, 68, 69 };
    print(c, 5);
}
```

```
65      66      67      68      69
65.1    66.2    67.3    68.4    69.5
65      66      67      68      69
```

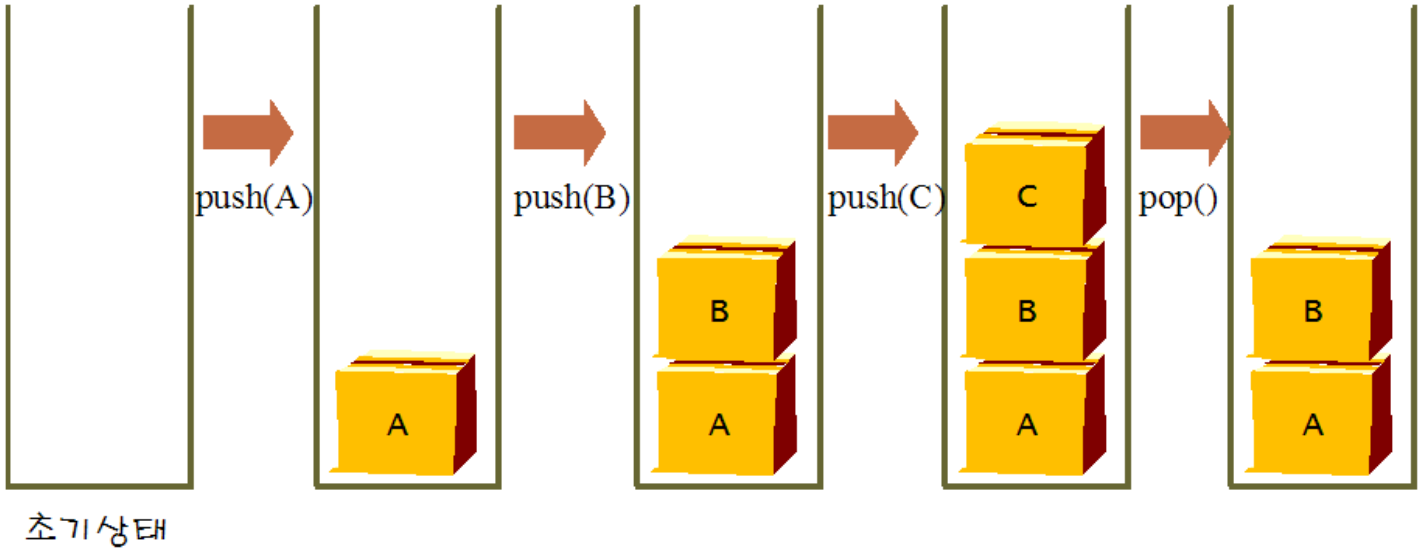
예제: 스택

- 스택(stack): 후입 선출(LIFO:Last-In First-Out) 자료 구조

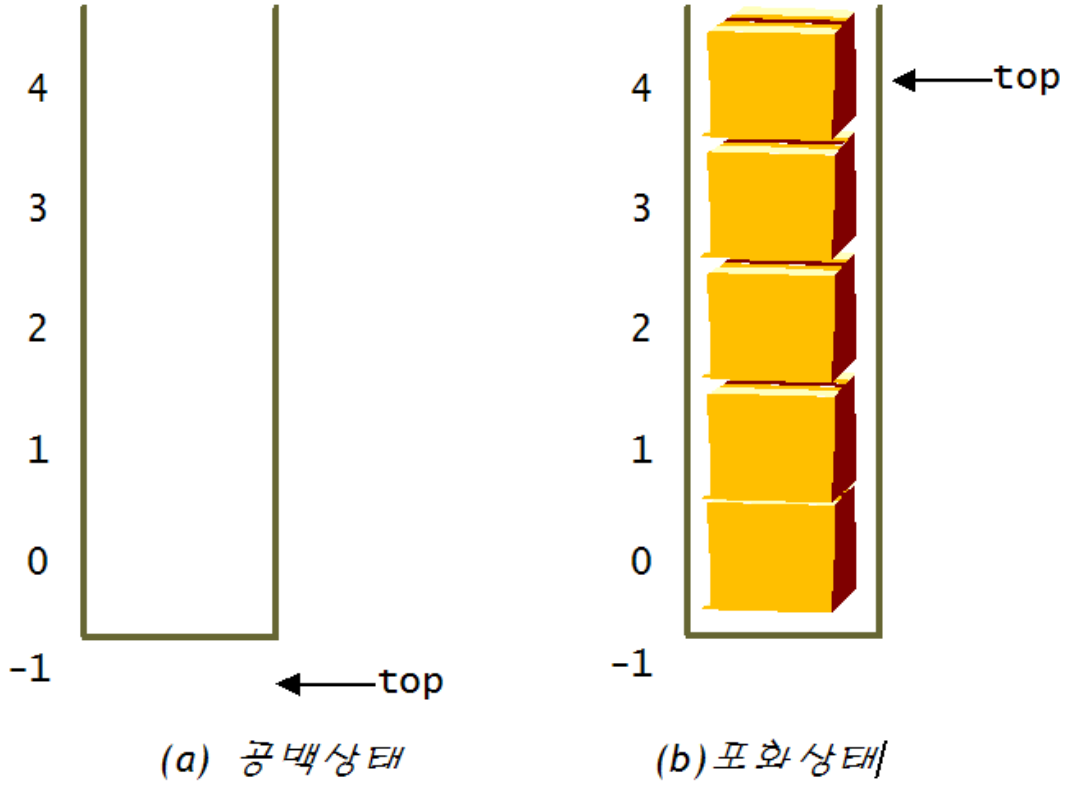


스택의 연산들

- is_empty(s) ::= 스택이 비어있는지를 검사한다.
- is_full(s) ::= 스택이 가득 찼는가를 검사한다.
- push(s, e) ::= 스택의 맨 위에 요소 e를 추가한다.
- pop(s) ::= 스택의 맨 위에 있는 요소를 삭제한다.



스택의 공백 상태와 포화 상태



(a) 공백상태 (b) 포화상태

isEmpty() , isFull()

isEmpty()

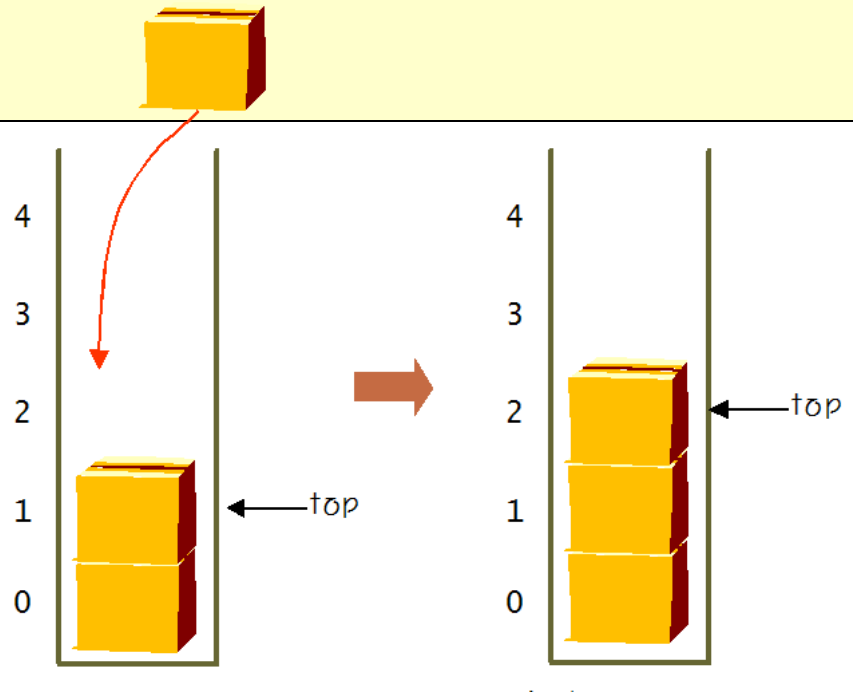
if *top* = -1
    then return TRUE
    else return FALSE

isFull()

if *top* = (MAX_STACK_SIZE-1)
    then return TRUE
    else return FALSE

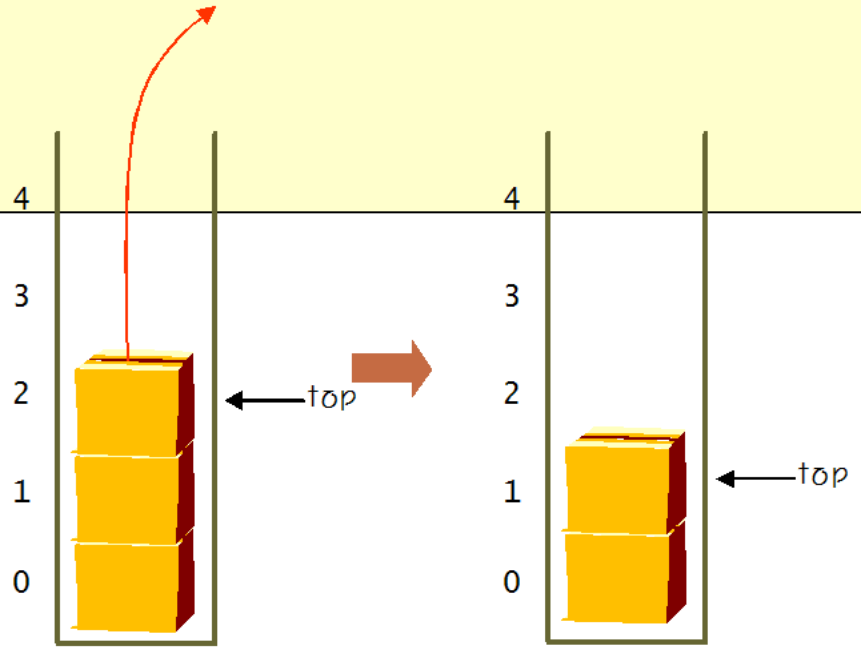
push()

```
push(x)

if isFull()
    then error "overflow"
    else top←top+1
         stack[top]←x
```



pop()

```
pop(x)

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



Stack 구현

```
#include <iostream>
using namespace std;
// 예외 처리를 위한 클래스
class FullStack
{};
// 예외 처리를 위한 클래스
class EmptyStack
{};
```

```
template <class T>
class Stack {
private:
    T*  s;
    int    size;
    int    top;
public:
    Stack(int n = 100) : size(n), top(-1)
    {
        s = new T[size];
    }
    ~Stack() { delete[]s; }
    void  push(T v);
    T pop();
    bool  isEmpty() const { return top == -1; }
    bool  isFull() const { return top == size - 1; }
};
```

Stack 구현

```
template< typename T >
void Stack< T >::push(T v)
{
    if (isFull())
        throw FullStack();
    s[++top] = v;
}

template< typename T >
T Stack< T >::pop()
{
    if (isEmpty())
        throw EmptyStack();
    return s[top--];
}
```

```
int main()
{
    Stack<int> s;  // 크기가 100인 정수형 스택
    s.push(100);
    s.push(200);
    s.push(300);
    s.push(400);
    cout << s.pop() << endl;
    cout << s.pop() << endl;
    cout << s.pop() << endl;
    cout << s.pop() << endl;
    return 0;
}
```

```
400
300
200
100
```

Q & A