

# 11장 표준 라이브러리와 표준 입출력

- ✦ C++ 표준 라이브러리의 구성
- ✦ C++ 입출력 클래스
- ✦ setf와 unsetf 멤버 함수를 이용한 입출력 형식 지정
- ✦ setf 이외의 멤버 함수를 이용한 입출력 형식 지정
- ✦ 입출력 조작자를 이용한 입출력 형식 지정
- ✦ 사용자 정의 입출력 조작자의 생성
- ✦ 문자 단위 입출력
- ✦ 줄 단위 입출력
- ✦ 입출력 스트림 상태
- ✦ string 클래스
- ✦ complex 클래스

# 1. C++ 표준 라이브러리의 구성

## ■ C++ 표준 라이브러리의 구성

### ■ 본 교재를 통해 소개할 라이브러리

범주	설명	예	관련 장/절
입출력	표준입출력 파일입출력	ostream ofstream	11.2~11.9절 12장
문자열	문자열 처리	string	11.10절
수치계산	수치계산 관련	complex	11.11절
컨테이너 클래스	다양한 자료구조	vector	13장
이터레이터	컨테이너 클래스 요소 포인터	iterator	
알고리즘	범용 함수	sort	
메모리	메모리 동적 할당	auto_ptr	15.1절
타입	타입 정보	typeid	15.4절

## 2. C++ 입출력 클래스

### # C 스타일의 입출력 : printf, scanf, fprintf, fscanf 등

- <cstdio> 헤더 파일에 포함되어 있음

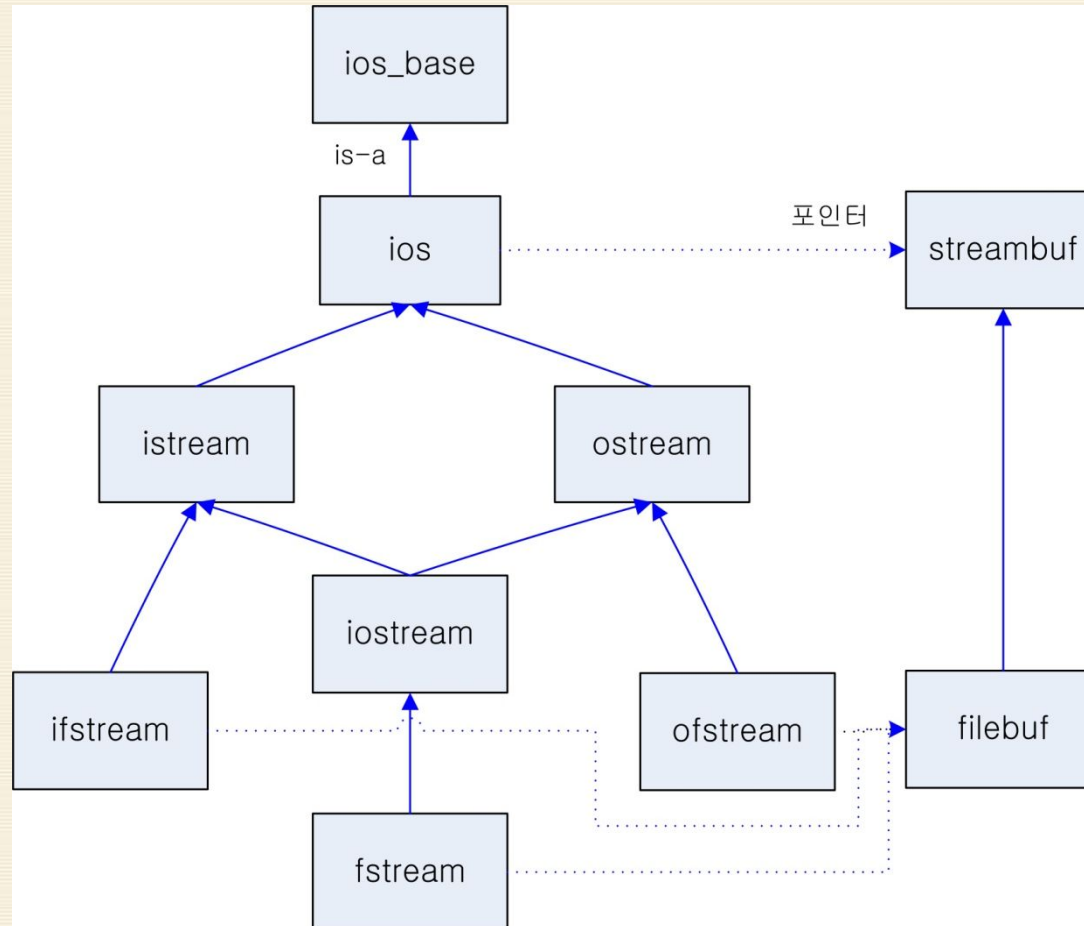
### # C++ 입출력 관련 클래스

사용 방법 동일  
향후로는 8비트를 대상으로 설명

템플릿 클래스	8비트 문자 기반 클래스	와이드 문자 기반 클래스
basic_streambuf	streambuf	wstreambuf
ios_base (비템플릿)	ios_base	ios_base
basic_ios	ios	wios
basic_istream	istream	wistream
basic_ostream	ostream	wostream
basic_filebuf	filebuf	wfilebuf
basic_ifstream	ifstream	wifstream
basic_ofstream	ofstream	wofstream
basic_fstream	fstream	wfstream

## 2. C++ 입출력 클래스

### ⌘ 입출력 관련 클래스들의 구성도



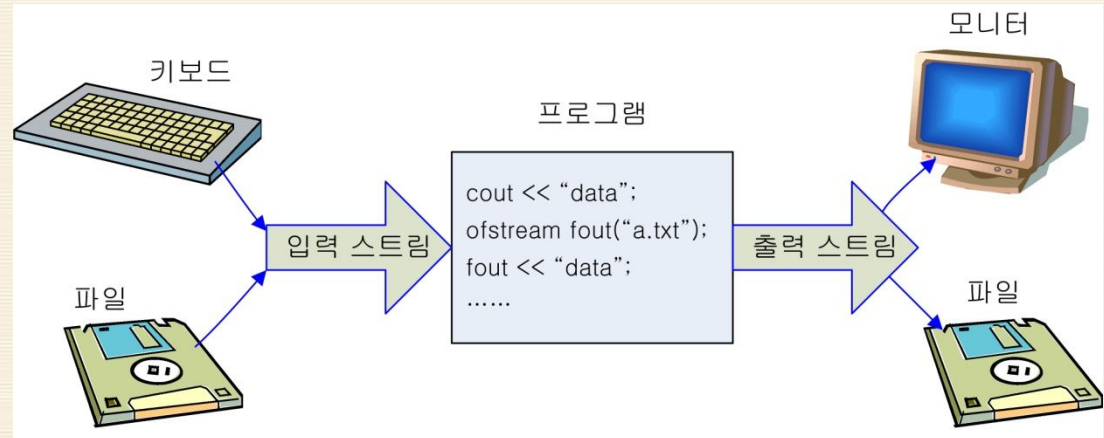
표준 입출력 클래스들  
**istream, ostream, iostream**

파일 입출력 클래스들  
**ifstream, ofstream, fstream**

## 2. C++ 입출력 클래스

### ✦ 입출력 메커니즘

- 표준 입출력(키보드, 모니터)과 파일 입출력(파일)의 사용 방법 유사



### ✦ 표준 입출력 객체

클래스	객체명	기능	연결 장치	대응 C 스트림
istream	cin	입력	키보드	stdin
ostream	cout	출력	모니터	stdout
ostream	cerr	오류 출력	모니터	stderr

- 표준 입출력 객체는 시스템에 의해 생성됨
- 입출력(표준, 파일) 객체의 복사 생성 및 대입 불가
  - ← 복사 생성자와 대입 연산자는 private 멤버로 포함되어 있음 (ios\_base)
- ostream, istream 구현 원리는 7.9절 참고 : 본 장에서는 사용 방법 설명



### 3. setf와 unsetf 멤버 함수를 이용한 입출력 형식 지정

#### ※ cout : setf와 unsetf를 이용한 출력 형식 지정 원리

- \_Fmtfl 변수(출력 형식 저장) 값 변경 : 각 서식의 비트 값을 1로 변경 또는 0으로 변경
- 각 서식의 의미를 열거형값으로 선언
  - ios\_base::hex

<cout 객체의 멤버 변수>

↓  
int \_Fmtfl

오른쪽	왼쪽	과학적	+표시	대문자	16진수	8진수	10진수
0	1	0	0	0	0→1	0	1→0

↑  
1로 변경

↑  
0으로 변경

cout.setf(4)

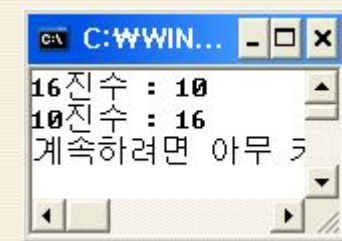
cout.unsetf(1)

#### ※ 16진수 출력 예

```
int main(void)
{
    cout.unsetf(ios_base::dec);    // 10진수 해제
    cout.setf(ios_base::hex);      // 16진수 설정
    cout << "16진수 : " << 16 << endl;

    cout.setf(ios_base::dec);      // 10진수 설정
    cout << "10진수 : " << 16 << endl;

    return 0;
}
```



### 3. setf와 unsetf 멤버 함수를 이용한 입출력 형식 지정

# fmtflags setf(fmtflags); 멤버 함수의 사용 (fmtflags는 int형과 동일)

```
int main(void)
{
    bool bTF = true;
    int i = 16;
    double d = 12.0;

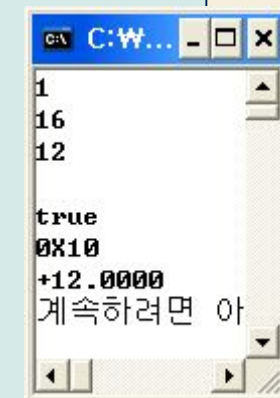
    cout << bTF << endl;
    cout << i << endl;
    cout << d << endl << endl;
    // 플래그 동시 설정
    cout.setf(ios_base::boolalpha | ios_base::showbase |
              ios_base::showpoint | ios_base::uppercase | ios_base::showpos);

    cout.unsetf(ios_base::dec);
    cout.setf(ios_base::hex);    // 16진수 설정

    cout << bTF << endl;
    cout << i << endl;
    cout << d << endl;

    return 0;
}
```

플래그 상수	설명	용도
boolalpha	bool 값을 true와 false로 표현	입출력
showbase	정수값 출력 시 진법 표시 접두어 사용	출력
showpoint	실수값 출력 시 소수점 표기	출력
uppercase	16진수 출력 시 X를 대문자로 표기 실수 과학적 표기 시 E를 대문자로 표기	출력
showpos	양수 앞에 + 부호 표기	출력
skipws	입력 시 공백 문자 무시 (디폴트 On)	입력



### 3. setf와 unsetf 멤버 함수를 이용한 입출력 형식 지정

#### ■ fmtflags setf(fmtflags, fmtflags); 멤버 함수의 사용

##### ■ 관련 그룹별 지정

첫번째 매개변수	두번째 매개변수	설명	용도
dec, oct, hex	basefield	10진수, 8진수, 16진수 표기	입출력
fixed, scientific	floatfield	실수의 소수점 표기와 과학적 표기	출력
left, right, internal	adjustfield	왼쪽 정렬, 오른쪽 정렬, 부호와 진법접두어는 왼쪽 정렬 이고 값은 오른쪽 정렬	출력

```
int main(void)
{
    bool bTF = true;
    int i = 16;
    double d = 12.0;
```

```
    cout << bTF << endl;
    cout << i << endl;
    cout << d << endl << endl;
```

```
    cout.setf(ios_base::boolalpha | ios_base::showbase |
              ios_base::showpoint | ios_base::uppercase | ios_base::showpos);
```

```
    cout.setf(ios_base::hex, ios_base::basefield);
    cout.setf(ios_base::scientific, ios_base::floatfield);
```

```
    cout << bTF << endl;
    cout << i << endl;
    cout << d << endl;
```

```
    return 0;
```

```
}
```

16진수 설정

과학적 표기법





### 3. setf와 unsetf 멤버 함수를 이용한 입출력 형식 지정

#### ✦ setf 함수의 반환값

- 이전 플래그 변수(\_Fmtfl)의 값 → 각 플래그의 상태를 알아볼 수 있음

#### ✦ 플래그 지정 없이 플래그 변수 값을 알아오는 함수

- fmtflags flags();

```
int main(void)
{
    ios_base::fmtflags flag = cout.flags();

    if (flag & ios_base::hex)    // 16진수 설정 여부
        cout << "hex on" << endl;
    else
        cout << "hex off" << endl;

    cout.setf(ios_base::hex, ios_base::basefield);

    flag = cout.flags();

    if (flag & ios_base::hex)
        cout << "hex on" << endl;
    else
        cout << "hex off" << endl;

    return 0;
}
```

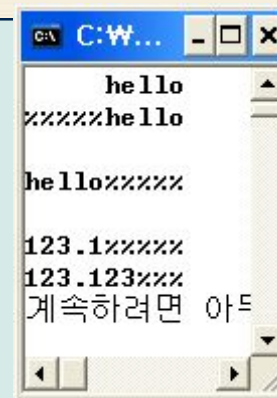
## 4. setf 이외의 멤버 함수를 이용한 출력 형식 지정

### ▣ 출력 형식 지정을 위한 ostream 멤버 함수

함수	설명	디폴트값
int width(int i)	최소 필드 너비 조정 조정 후 첫 번째 출력 후에는 디폴트 값으로 자동 환원	출력 내용 과 동일
char fill(char c)	필드 내의 공백 자리에 채워질 문자 설정	공백
int precision(int p)	실수 출력 시 출력되는 총 자릿수, 출력 형식이 fixed 또는 scientific이라면 소수점 이하 자릿수	6

```
int main(void)
{
    cout.width(10);
    cout << "hello" << endl;
    cout.fill('%');
    cout.width(10);
    cout << "hello" << endl << endl;

    cout.setf(ios::left, ios::adjustfield);
    cout.width(10);
    cout << "hello" << endl << endl;
}
```



```
cout.width(10);
cout.precision(4);
cout << 123.1234567 << endl;
cout.width(10);
cout.precision(6);
cout << 123.1234567 << endl;

return 0;
}
```

## 5. 입출력 조작자를 이용한 입출력 형식 지정

### ⌘ 입출력 형식 지정 방법

- ostream, istream 클래스의 멤버 함수 사용 : setf, unsetf, width 등
- 입출력 조작자(io manipulator) 사용 : 동일한 효과를 낼 수 있음

### ⌘ 입출력 조작자의 사용 방법 및 수행 원리

- <<, >> 입출력 연산자와 함께 사용하게 됨
  - `cout << oct << 100 << hex << 100 << endl;`
- 수행 원리 : 내부적으로는 결국 ostream, istream 의 멤버 함수가 수행됨
  - `cout << hex`의 경우
  - 변환 수행 : `cout.operator<<(hex);`
  - 함수 호출 : `hex(cout);` // hex라는 함수가 수행됨
  - 수행 내용 : `cout.setf(ios_base::hex);` // hex 함수의 수행 내용

### ⌘ setf, unsetf에 해당하는 입출력 조작자

- setiosflags, resetiosflags
  - `cout << resetiosflags(ios_base::dec) << setiosflags(ios_base::hex) << 16 << endl;`

## 5. 입출력 조작자를 이용한 입출력 형식 지정

### ▣ 입출력 조작자의 종류

입출력 조작자	대응 서식 플래그	설명	용도
boolalpha noboolalphs	setf(ios_base::boolalpha) unsetf(ios_base::boolalpha)	bool 값의 true, false 표현	입출력
showbase noshowbase	setf(ios_base::showbase) unsetf(ios_base::showbase)	진법 표시 접두어 사용	출력
showpoint noshowpoint	setf(ios_base::showpoint) unsetf(ios_base::showpoint)	소수점 표기	출력
uppercase nouppercase	setf(ios_base::uppercase) unsetf(ios_base::uppercase)	16진수 X, 과학적 표기 E 대문자 사용	출력
showpos noshowpos	setf(ios_base::showpos) unsetf(ios_base::showpos)	양수 앞에 + 부호 표기	출력
skipws noskipws	setf(ios_base::skipws) unsetf(ios_base::skipws)	입력 시 공백 문자 무시 (디폴트 값 : On)	입력
dec hex oct	setf(ios_base::dec, ios_base::basefield)	진수 표기	입출력
fixed scientific	setf(ios_base::fixed, ios_base::floatfield)	실수 표기	출력
left internal right	setf(ios_base::left, ios_base::adjustfield)	정렬 방법	출력

## 5. 입출력 조작자를 이용한 입출력 형식 지정

### ⌘ 입출력 조작자의 종류 (계속)

입출력 조작자	대응 서식 플래그	설명	용도
setw(int)	width(int)	필드 너비 조정, 이후 한 번의 출력 후 디폴트로 환원됨	출력
setfill(char)	fill(char)	공백 자리 채움 문자 지정	출력
setprecision(int)	precision(int)	실수 출력 자릿수 설정	출력
endl		newline 문자 출력, 스트림 비움	출력
flush		스트림을 비움	출력
setiosflags	setf		
resetiosflags	unsetf		

```
#include <iostream>
#include <iomanip>           // setfill, setw
using namespace std;

int main(void)
{
    cout << hex << 100 << endl;    // 16진수 출력
    cout << oct << 10 << endl;     // 8진수 출력

    cout << setfill('X') << 100 << setw(10) << endl; // 채움문자, 필드 설정
    cout << 100 << " hi " << endl;

    return 0;
}
```





## 6. 사용자 정의 입출력 조작자의 생성

### ⌘ 입출력 조작자(결국 함수)의 형태

- `ostream &FuncName(ostream &);` // 출력 조작자
- `istream &FuncName(istream &);` // 입력 조작자

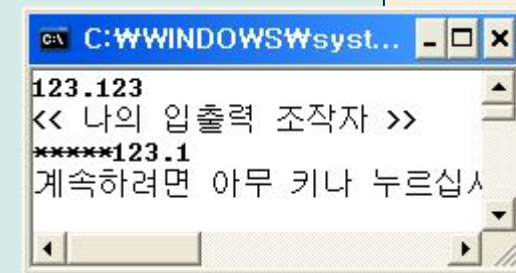
### ⌘ 사용자 정의 조작자의 예

```
ostream &MySetup(ostream &os)      // 사용자 정의 출력 조작자
{
    os << "<< 나의 입출력 조작자 >>" << endl;
    os.width(10);
    os.precision(4);
    os.fill('*');

    return os;
}

int main(void)
{
    cout << 123.123456 << endl;
    cout << MySetup << 123.123456 << endl;

    return 0;
}
```



## 7. 문자 단위 입출력

### ✦ istream의 문자 입력 함수

- `int get(void);`
- `istream &get(char &);`      // 연속적인 get 함수 적용 가능

### ✦ ostream의 문자 출력 함수

- `ostream &put(char);`

### ✦ 예 : 키보드 입력을 그대로 화면에 출력

```
int main(void)
{
    char ch;

    cin.get(ch);           // 문자 하나 입력

    while (!cin.eof()) {
        cout.put(ch);      // 문자 하나 출력
        ch = cin.get();    // 문자 하나 입력
    }

    return 0;
}
```



파일 끝 표시 : **Ctrl+z**

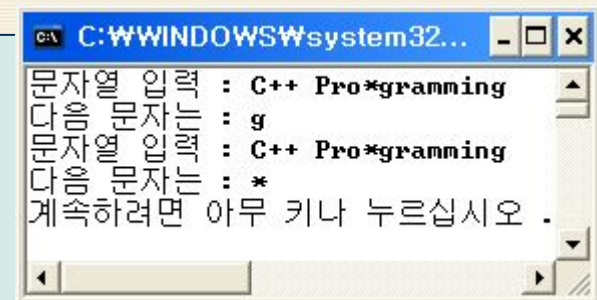
## 8. 줄 단위 입력

✦ istream : 줄 단위 문자열 입력을 위한 멤버 함수

- istream &get(char \*, int, char = 'Wn');
  - 최대 개수(int) 만큼 입력을 받아들이되 종료 문자('Wn')가 나타나면 입력 종료
  - 입력 후 종료 문자를 입력 스트림에 남겨둠
- istream &getline(char \*, int, char = 'Wn');
  - 입력 후 종료 문자를 입력 스트림으로부터 제거

```
char str[80];  
cin.get(str, 80);  
cin.getline(str, 80);
```

```
int main(void)  
{  
    char str[80];  
  
    cout << "문자열 입력 : ";  
    cin.getline(str, 80, '*');  
    cout << "다음 문자는 : " << (char) cin.get() << endl;  
  
    while ((cin.get()) != 'Wn');    // 그 줄의 나머지를 읽어들이м  
  
    cout << "문자열 입력 : ";  
    cin.get(str, 80, '*');          // '*' 문자 전까지 입력  
    cout << "다음 문자는 : " << (char) cin.get() << endl;  
  
    return 0;  
}
```



## 9. 입출력 스트림 상태

### ▣ 입출력 수행에 따른 현재 상태 저장

- ios\_base 클래스 내에 상태를 저장하는 변수가 있음

➢ iostate \_State; // iostate는 int와 동일

✓ 실제 유의미한 비트는 하위 3비트

- 입출력 스트림 상태의 종류

상태	열거값	설명	접근 함수
goodbit	0	에러가 발생하지 않았음 eofbit, failbit, badbit 모두 0인 경우	good()
eofbit	1	입력스트림에서 파일의 끝에 도달	eof()
failbit	2	치명적이지 않은 입출력 에러 발생 <ul style="list-style-type: none"><li>▪ 지정한 타입의 값을 읽을 수 없음</li><li>▪ 접근할 수 없는 파일 읽기</li><li>▪ 쓰기 방지된 디스켓에 쓰기</li></ul>	fail()
badbit	4	치명적인 입출력 에러 발생 <ul style="list-style-type: none"><li>▪ 복구 불가능한 에러</li></ul>	bad()

- 현재 상태가 fail인지 알고 싶다면 : `if (cin.fail()) { ... }`
- 스트림 상태를 goodbit(0)으로 복원하는 방법 : `cin.clear();`
- 현재 스트림 상태값(\_State)을 읽어오는 함수 : `iostate rdstate() const;`

## 9. 입출력 스트림 상태

✦ 예 : 원하는 타입의 데이터를 읽지 못한 경우 failbit의 값 확인

```
int main(void) {
    int a;

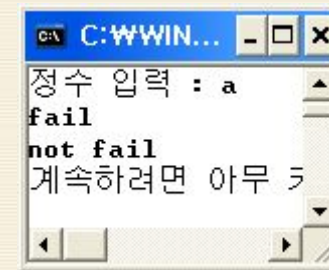
    cout << "정수 입력 : ";
    cin >> a;

    if (cin.fail())
        cout << "fail" << endl;
    else
        cout << "not fail" << endl;

    cin.clear();

    if (cin.fail())
        cout << "fail" << endl;
    else
        cout << "not fail" << endl;

    return 0;
}
```



다음 장에서 배울 파일입출력을 위해  
사용 가능

→ 지금까지 배운 내용(서식 지정, 입출력)  
모두 파일입출력에 사용 가능

클래스 객체에 대한 출력 연산자 오버로딩  
및 입력 연산자 오버로딩 → 7.9절 참고



## 10. string 클래스

### ⌘ (연습 문제 7.14) CString 클래스 만들기

- 내부적으로 char 포인터와 문자열 길이 보관
- 생성자, 소멸, 복사생성자, 대입 연산자, +, <<, +=, ==, >> 연산자 오버로딩 필요

### ⌘ 표준 C++에서 제공하는 문자열 처리 클래스 → string

- <string> 헤더 파일에 포함되어 있음
- 주요 기능 (교재 [표 11.9] 참고)
  - = : 대입 연산
  - +, += : 문자열 결합
  - ==, !=, <, >, <=, >= : 동등 및 대소 비교
  - >>, << : 입출력 연산
  - [], append, insert, erase, replace, find, rfind, compare, swap

## 10. string 클래스

### # insert와 swap 함수의 사용 예

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

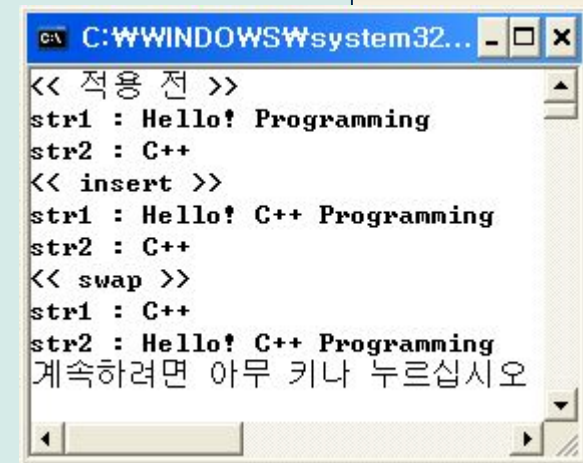
void Print(string &s1, string &s2, string title)
{
    cout << title << endl;
    cout << "str1 : " << s1 << endl;
    cout << "str2 : " << s2 << endl;
}

int main(void)
{
    string str1 = "Hello! Programming";
    string str2 = "C++ ";
    Print(str1, str2, "<< 적용 전 >>");

    str1.insert(7, str2);           // 7과 8번째 원소 사이에 삽입
    Print(str1, str2, "<< insert >>");

    str1.swap(str2);               // 문자열 교환
    Print(str1, str2, "<< swap >>");

    return 0;
}
```



```
<< 적용 전 >>
str1 : Hello! Programming
str2 : C++
<< insert >>
str1 : Hello! C++ Programming
str2 : C++
<< swap >>
str1 : C++
str2 : Hello! C++ Programming
계속하려면 아무 키나 누르십시오
```

# 11. complex 클래스

■ complex 클래스 : complex 헤더 파일에 포함되어 있음

- 복소수를 표현하는 클래스 :  $a + bi$  (a:실수부, b:허수부)
- 템플릿 클래스로 구현
- 주요 기능

기능	연산자	의미( $X = a + bi$ , $Y = c + di$ )
덧셈	+	$X + Y = (a + c) + (b + d)i$
뺄셈	-	$X - Y = (a - c) + (b - d)i$
곱셈	*	$X * Y = (ac - bd) + (ad + bc)i$
나눗셈	/	$X / Y = \{(ac + bd)/(c^2 + d^2)\} + \{(bc - ad)/(c^2 + d^2)\}i$
대입	=	
상등 비교	==, !=	
입출력	>>, <<	

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

```
int main(void)
{
```

```
    complex<double> comp1(1.0, 2.0);
    complex<double> comp2(3.0, 4.0);
```

```
    cout << "+ : " << comp1 + comp2 << endl;
    cout << "- : " << comp1 - comp2 << endl;
    cout << "* : " << comp1 * comp2 << endl;
    cout << "/ : " << comp1 / comp2 << endl;
```

```
    return 0;
```

```
}
```

