

Chapter 02. C++ 프로그래밍의 기본



학습 목표

1. C++ 프로그램의 기본 요소를 이해한다.
2. namespace와 std:: 등 이름 공간을 사용할 수 있다.
3. 키 입력 받는 방법을 안다.
4. 문자열을 입력 받는 방법을 안다.
5. 간단한 C++ 프로그램을 작성할 수 있다.
6. #include <iostream>의 실체에 대해 이해한다.

예제 2-1: 기본적인 C++ 프로그램

```
/*
    소스: SimpleC++.cpp
    cout과 << 연산자를 이용하여 화면에 출력한다.
*/

#include <iostream> // cout과 << 연산자 포함

// C++ 프로그램은 main() 함수에서부터 실행을 시작한다.
int main() {
    std::cout << "Hello\n"; // 화면에 Hello를 출력하고 다음 줄로 넘어감
    std::cout << "첫 번째 맛보기입니다.";
    return 0; // main() 함수가 종료하면 프로그램이 종료됨
}
```

```
Hello
첫 번째 맛보기입니다.
```

주석문과 main() 함수

- 주석문
 - 개발자가 자유롭게 붙인 특이 사항의 메모, 프로그램에 대한 설명
 - 프로그램의 실행에 영향을 미치지 않음
 - 여러 줄 주석문 - /* ... */
 - 한 줄 주석문 - //를 만나면 이 줄의 끝까지 주석으로 처리
- main() 함수
 - C++ 프로그램의 실행을 시작하는 함수
 - main() 함수가 종료하면 C++ 프로그램 종료
 - main() 함수의 C++ 표준 모양

```
int main() { // main()의 리턴 타입 int
    .....
    return 0; // 0이 아닌 다른 값으로 리턴 가능
}
```

```
void main() { // 표준 아님
    .....
}
```

- main()에서 return문 생략 가능

```
int main() {
    .....
    // return 0; // 개발자의 편리를 위해 return 문 생략 가능
}
```

#include <iostream>

- #include <iostream>
 - 전처리기(C++ Preprocessor)에게 내리는 지시
 - <iostream> 헤더 파일을 컴파일 전에 소스에 확장하도록 지시
- <iostream> 헤더 파일
 - 표준 입출력을 위한 클래스와 객체, 변수 등이 선언됨
 - ios, istream, ostream, iostream 클래스 선언
 - cout, cin, <<, >> 등 연산자 선언

```
#include <iostream>
```

```
....
```

```
std::cout << "HelloWn";
```

```
std::cout << "첫 번째 맛보기입니다.";
```

printf()는 잊어라!



C 언어에서 사용했던 printf를 더 이상 C++에서 사용하지 말기 바란다. printf()나 scanf() 등을 사용하여 구석기 시대로 회귀한다면, 더 이상 C++ 프로그래머로서의 미래는 없다.



화면 출력

■ cout과 << 연산자 이용 (← 11장)

```
std::cout << "Hello\n"; // 화면에 Hello를 출력하고 다음 줄로 넘어감  
std::cout << "첫 번째 맛보기입니다.";
```

■ cout 객체

- 스크린 출력 장치에 연결된 **표준** C++ 출력 스트림 객체
- <iostream> 헤더 파일에 선언
- std 이름 공간에 선언: **std::cout**으로 사용

■ << 연산자

- 스트림 삽입 연산자(stream insertion operator)
 - C++ 기본 산술 시프트 연산자(<<)가 스트림 삽입 연산자로 재정의됨 (← 7장)
 - ostream 클래스에 구현됨
 - 오른쪽 피연산자를 왼쪽 스트림 객체에 삽입
 - cout 객체에 연결된 화면에 출력
- 여러 개의 << 연산자로 여러 값 출력

```
std::cout << "Hello\n" << "첫 번째 맛보기입니다.";
```

<< 연산자 활용

- 문자열 및 기본 타입의 데이터 출력

- bool, char, short, int, long, float, double 타입 값 출력

```
int n=3;  
char c='#';  
std::cout << c << 5.5 << '-' << n << "hello" << true;
```

#5.5-3hello1

- 연산식뿐 아니라 함수 호출도 가능

```
std::cout << "n + 5 =" << n + 5;  
std::cout << f(); // 함수 f()의 리턴값을 출력한다.
```

- 다음 줄로 넘어가기

- '\n'이나 endl 조작자 사용 (← 11장)

```
std::cout << "Hello" << '\n';  
std::cout << "Hello" << std::endl;
```


예제 2-2 cout과 <<를 이용한 화면 출력

```
#include <iostream>
```

```
double area(int r); // 함수의 원형 선언
```

```
double area(int r) { // 함수 구현  
    return 3.14*r*r; // 반지름 r의 원면적 리턴  
}
```

```
int main() {  
    int n=3;  
    char c='#';  
    std::cout << c << 5.5 << '-' << n << "hello" << true << std::endl;  
    std::cout << "n + 5 = " << n + 5 << '\n';  
    std::cout << "면적은 " << area(n); // 함수 area()의 리턴 값 출력  
}
```

true는 1
로 출력됨

```
#5.5-3hello1
```

```
n + 5 = 8
```

```
면적은 28.26
```

이름 충돌 사례



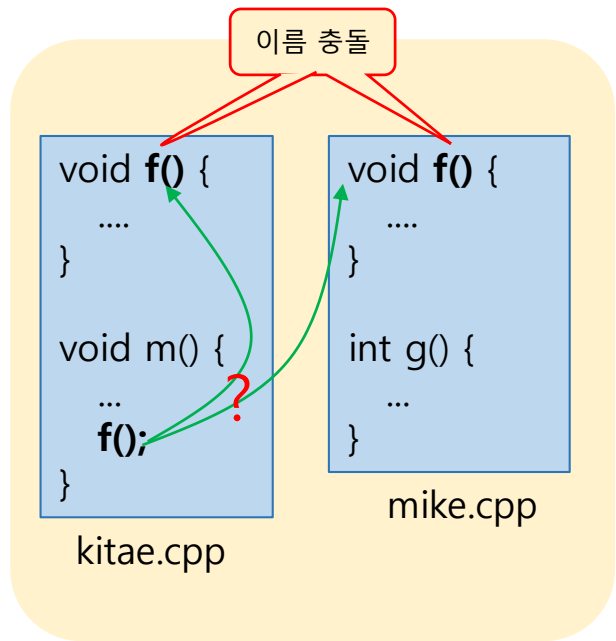
우리 아파트에 여러 명의 마이클이 산다.
마이클을 부를 때, 1동::마이클, 2동::마이클로 부른다.

namespace 개념

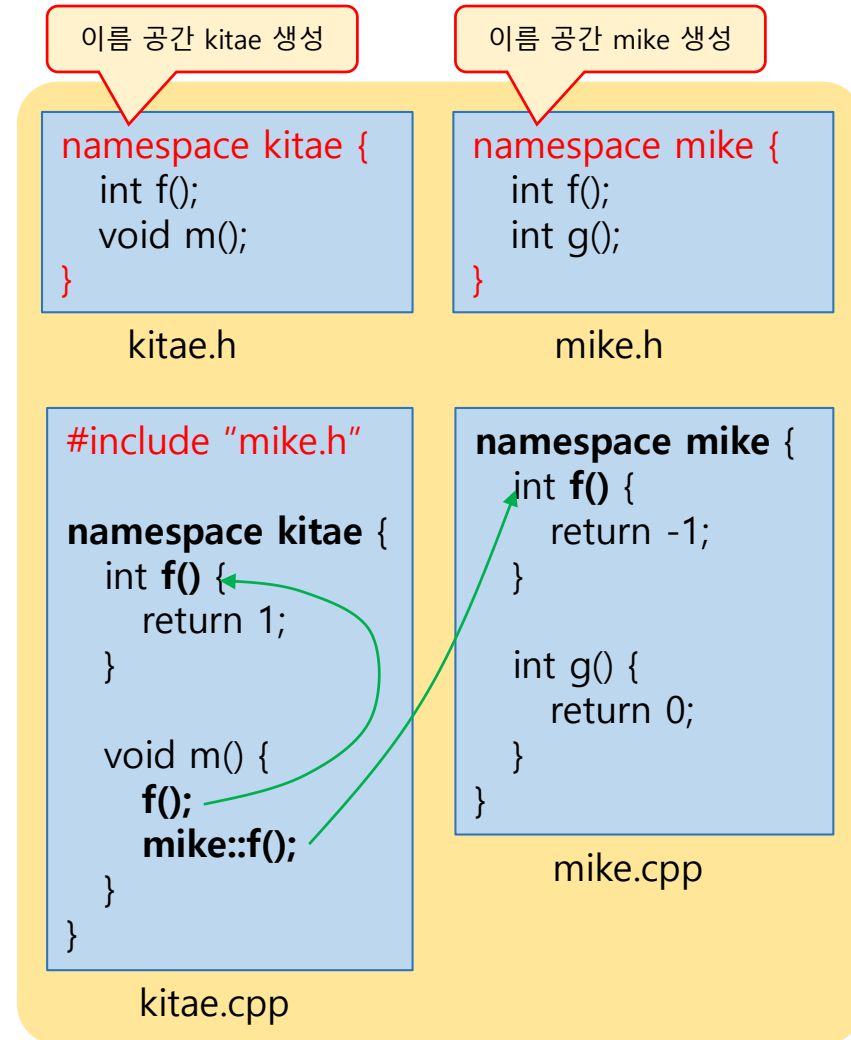
- 이름(identifier) 충돌이 발생하는 경우
 - 여러 명이 서로 나누어 프로젝트를 개발하는 경우
 - 오픈 소스 혹은 다른 사람이 작성한 소스나 목적 파일을 가져와서 컴파일 하거나 링크하는 경우
 - 해결하는데 많은 시간과 노력이 필요
- namespace 키워드
 - 이름 충돌 해결
 - 2003년 새로운 C++ 표준에서 도입
 - 개발자가 자신만의 이름 공간을 생성할 수 있도록 함
 - 이름 공간 안에 선언된 이름은 다른 이름공간과 별도 구분
- 이름 공간 생성 및 사용

```
namespace kitae { // kitae 라는 이름 공간 생성
..... // 이 곳에 선언된 모든 이름은 kitae 이름 공간에 생성된 이름
}
```

- 이름 공간 사용
 - 이름 공간 :: 이름



(a) kitae와 mike에 의해 작성된 소스를 합치면 f() 함수의 이름 충돌. 컴파일 오류 발생



(b) 이름 공간을 사용하여 f() 함수 이름의 충돌 문제 해결

std:: 란?

- std
 - C++ 표준에서 정의한 **이름 공간(namespace)** 중 하나
 - <iostream> 헤더 파일에 선언된 모든 이름: std 이름 공간 안에 있음
 - cout, cin, endl 등
 - std 이름 공간에 선언된 이름을 접근하기 위해 std:: 접두어 사용
 - std::cout, std::cin, std::endl

- std:: 생략

- using 지시어 사용

```
using std::cout; // cout에 대해서만 std:: 생략
```

```
.....  
cout << "Hello" << std::endl; // std::cout에서 std:: 생략
```

std:: 생략

```
using namespace std; // std 이름 공간에 선언된 모든 이름에 std:: 생략
```

```
.....  
cout << "Hello" << endl; // std:: 생략
```

std:: 생략

std:: 생략

#include <iostream>과 std

- <iostream>이 통째로 std 이름 공간 내에 선언
 - <iostream> 헤더 파일을 사용하려면 다음 코드 필요

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

예제 2-3 C++ 프로그램에서 키 입력 받기

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

int main() {
    cout << "너비를 입력하세요>>";

    int width;
    cin >> width; // 키보드로부터 너비를 읽어 width 변수에 저장

    cout << "높이를 입력하세요>>";

    int height;
    cin >> height; // 키보드로부터 높이를 읽어 height 변수에 저장

    int area = width*height; // 사각형의 면적 계산
    cout << "면적은 " << area << "Wn"; // 면적을 출력하고 다음 줄로 넘어감
}
```

```
너비를 입력하세요>>3
높이를 입력하세요>>5
면적은 15
```

cin과 >> 연산자를 이용한 키 입력

- cin
 - 표준 입력 장치인 키보드를 연결하는 C++ 입력 스트림 객체
- >> 연산자
 - 스트림 추출 연산자(stream extraction operator)
 - C++ 산술 시프트 연산자(>>)가 <iostream> 헤더 파일에 스트림 추출 연산자로 재정의됨
 - 입력 스트림에서 값을 읽어 변수에 저장
 - 연속된 >> 연산자를 사용하여 여러 값 입력 가능

```
cout << "너비와 높이를 입력하세요>>";  
cin >> width >> height;  
cout << width << 'Wn' << height << 'Wn';
```

```
너비와 높이를 입력하세요>>23 36  
23  
36
```

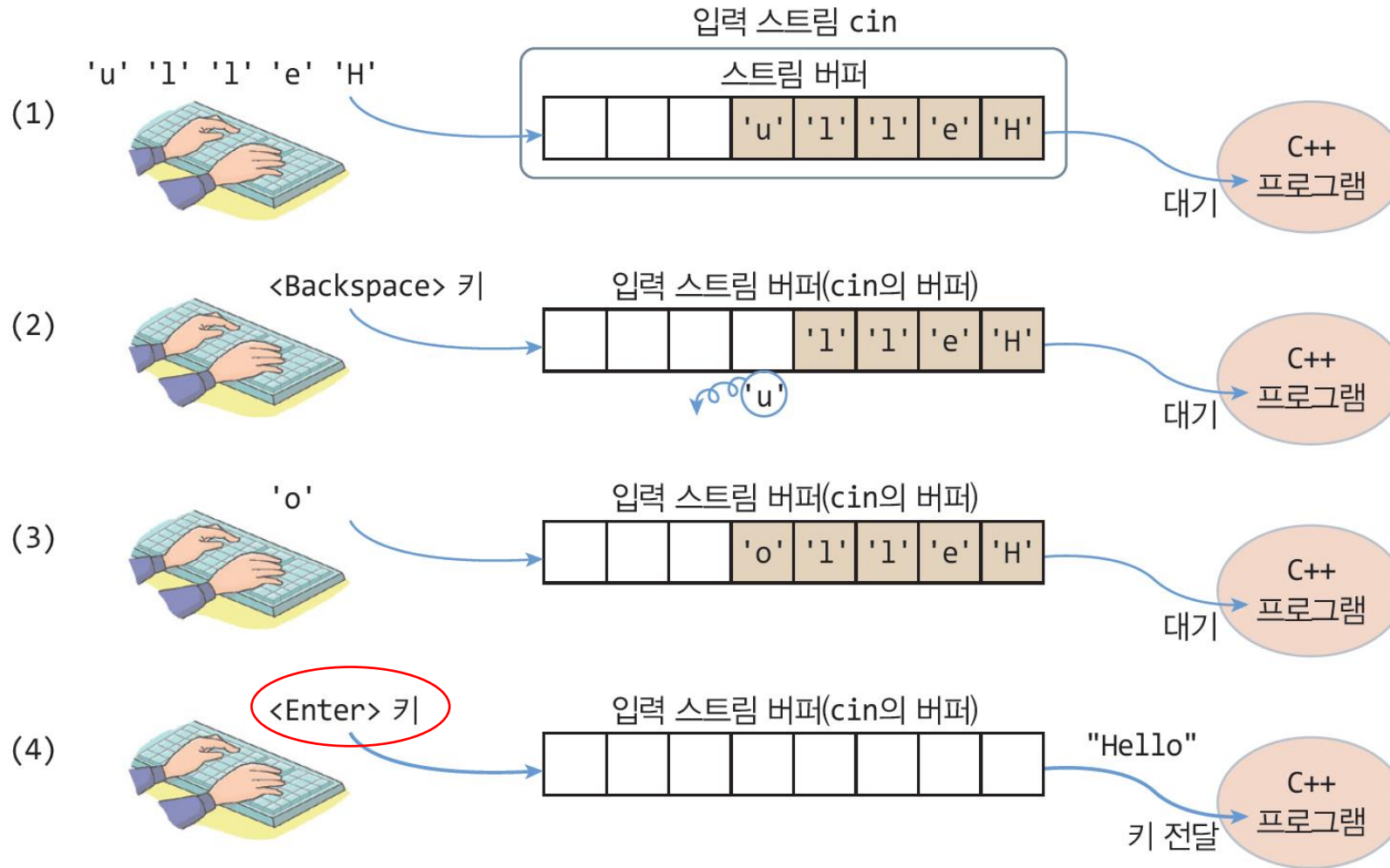
width에
입력

height에
입력

<Enter> 키를 칠 때 변수에 값 전달

- cin의 특징
 - 입력 버퍼를 내장하고 있음
 - <Enter>키가 입력될 때까지 입력된 키를 입력 버퍼에 저장
 - 도중에 <Backspace> 키를 입력하면 입력된 키 삭제
- >> 연산자
 - <Enter>키가 입력되면 비로소 cin의 입력 버퍼에서 키 값을 읽어 변수에 전달

cin으로부터 키 입력 받는 과정(11.1절)



실행문 중간에 변수 선언

■ C++의 변수 선언

- 변수 선언은 아무 곳이나 가능

실행문 중간
에 변수 선언

```
int width;  
cin >> width; // 키보드로부터 너비를 읽는다.  
  
cout << "높이를 입력하세요>>";  
  
int height;  
cin >> height; // 키보드로부터 높이를 읽는다.  
  
// 너비와 높이로 구성되는 사각형의 면적을 계산한다.  
int area = width*height;  
cout << "면적은 " << area << "Wn"; // 면적을 출력하고 한 줄 뺐다.
```

- C++ 변수 선언 방식의 장점
 - 변수를 사용하기 직전 선언함으로써 변수 이름에 대한 타이핑 오류 줄임
- C++ 변수 선언 방식의 단점
 - 선언된 변수를 일괄적으로 보기 힘들
 - 코드 사이에 있는 변수 찾기 어려움

타이핑 오류 가능성 해소

- 선언부에 모든 변수를 선언하는 경우, 타이핑 오류 가능

```
int time, timer;  
...  
timer = 5; // time에 5을 저장하려다 timer로 잘못 입력. 컴파일 오류 발생하지 않음  
           // 그러나 잘못된 실행 결과 발생  
....  
timer = 3;
```

- 변수 사용 전에 변수를 선언하면, 타이핑 오류 사전 발견

컴파일 오류

```
int time;  
timer = 5; // time에 5을 저장하려다 timer로 잘못 입력. 컴파일 오류 발생  
....  
int timer;  
timer = 3;
```

C++ 문자열

- C++의 문자열 표현 방식 : 2가지
 - C-스tring 방식 - '₩0'로 끝나는 문자 배열

C-스tring 문자열
char name1[6] = {'G', 'r', 'a', 'c', 'e', '₩0'}; // name1은 문자열 "Grace"
단순 문자 배열
char name2[5] = {'G', 'r', 'a', 'c', 'e'}; // name2는 문자열이 아니고 단순 문자 배열

char name5[10] = "Grace";

name5[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
'G'	'r'	'a'	'c'	'e'	'₩0'	'₩0'	'₩0'	'₩0'	'₩0'

"Grace" 문자열 '₩0'로 초기화

- string 클래스 이용
 - <string> 헤더 파일에 선언됨
 - 다양한 멤버 함수 제공, 문자열 비교, 복사, 수정 등

C-STRING 방식으로 문자열 다루기

- C-STRING으로 문자열 다루기
 - C 언어에서 사용한 함수 사용 가능
 - strcmp(), strlen(), strcpy() 등
 - <cstring> 이나 <string.h> 헤더 파일 include

```
#include <cstring> 또는  
#include <string.h>  
...  
int n = strlen("hello");
```

- <cstring> 헤더 파일을 사용하는 것이 바람직함
 - <cstring> 이 C++ 표준 방식

cin을 이용한 문자열 입력

■ 문자열 입력

```
char name[6]; // 5 개의 문자를 저장할 수 있는 char 배열  
cin >> name; // 키보드로부터 문자열을 읽어 name 배열에 저장한다.
```

Grace

키 입력

name [0] [1] [2] [3] [4] [5]

'G'	'r'	'a'	'c'	'e'	'\0'
-----	-----	-----	-----	-----	------

"Grace" 문자열

예제 2-4 키보드에서 문자열 입력 받고 출력

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

int main() {
    cout << "이름을 입력하세요>>";

    char name[11]; // 한글은 5개 글자, 영문은 10까지 저장할 수 있다.
    cin >> name; // 키보드로부터 문자열을 읽는다.

    cout << "이름은 " << name << "입니다\n"; // 이름을 출력한다.
}
```

이름을 입력하세요>>마이클
이름은 마이클입니다

빈 칸 없이 키 입력해야 함

이름을 입력하세요>>마 이 클
이름은 마입니다

빈 칸을 만나면 문자
열 입력 종료

예제 2-5 C-스트링을 이용하여 암호가 입력되면 프로그램을 종료하는 예

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

int main() {
    char password[11];
    cout << "프로그램을 종료하려면 암호를 입력하세요." << endl;
    while(true) {
        cout << "암호>>";
        cin >> password;
        if(strcmp(password, "C++") == 0) {
            cout << "프로그램을 정상 종료합니다." << endl;
            break;
        }
        else
            cout << "암호가 틀립니다~~" << endl;
    }
}
```

strcmp() 함수를 사용
하기 위한 헤더 파일

프로그램을 종료하려면 암호를 입력하세요.

암호>>Java

암호가 틀립니다~~

암호>>C

암호가 틀립니다~~

암호>>C++

프로그램을 정상 종료합니다.

빈 칸 없이 키 입력해야 함

cin.getline()으로 공백이 낀 문자열 입력

- 공백이 낀 문자열을 입력 받는 방법
- cin.getline(char buf[], int size, char delimiterChar)
 - buf에 최대 size-1개의 문자 입력. 끝에 '\0' 붙임
 - delimiterChar를 만나면 입력 중단. 끝에 '\0' 붙임
 - delimiterChar의 디폴트 값은 '\n'(<Enter>키)

```
char address[100];  
cin.getline(address, 100, '\n');
```

최대 99개의 문자를 읽어 address 배열에 저장. 도중에 <Enter> 키를 만나면 입력 중단

사용자가 'Seoul Korea<Enter>'를 입력할 때,

address[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [99]

'S'	'e'	'o'	'u'	'l'	' '	'K'	'o'	'r'	'e'	'a'	'\0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	-----	-----

"Seoul Korea" 문자열

예제 2-6 cin.getline()을 이용한 문자열 입력

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

int main() {
    cout << "주소를 입력하세요>>";

    char address[100];
    cin.getline(address, 100, '\n'); // 키보드로부터 주소 읽기

    cout << "주소는 " << address << "입니다\n"; // 주소 출력
}
```

주소를 입력하세요>>컴퓨터시 프로그램구 C++동 스트링 1-1
주소는 컴퓨터시 프로그램구 C++동 스트링 1-1입니다

빈칸이 있어도 <Enter> 키가 입력
될 때까지 하나의 문자열로 인식

C++에서 문자열을 다루는 string 클래스

- string 클래스 (← 4장)
 - C++에서 강력 추천
 - C++ 표준 클래스
 - 문자열의 크기에 따른 제약 없음
 - string 클래스가 스스로 문자열 크기게 맞게 내부 버퍼 조절
 - 문자열 복사, 비교, 수정 등을 위한 다양한 함수와 연산자 제공
 - 객체 지향적
 - <string> 헤더 파일에 선언
 - #include <string> 필요
 - C-스트링보다 다루기 쉬움

예제 2-7 string 클래스를 이용한 문자열 입력 및 다루기

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

int main() {
    string song("Falling in love with you"); // 문자열 song
    string elvis("Elvis Presley"); // 문자열 elvis
    string singer; // 문자열 singer

    cout << song + "를 부른 가수는"; // + 로 문자열 연결
    cout << "(힌트 : 첫글자는 " << elvis[0] << ")?"; // [] 연산자 사용

    getline(cin, singer); // 문자열 입력
    if(singer == elvis) // 문자열 비교
        cout << "맞았습니다.";
    else
        cout << "틀렸습니다. " + elvis + "입니다." << endl; // +로 문자열 연결
}
```

빈칸을 포함하는
문자열 입력 가능

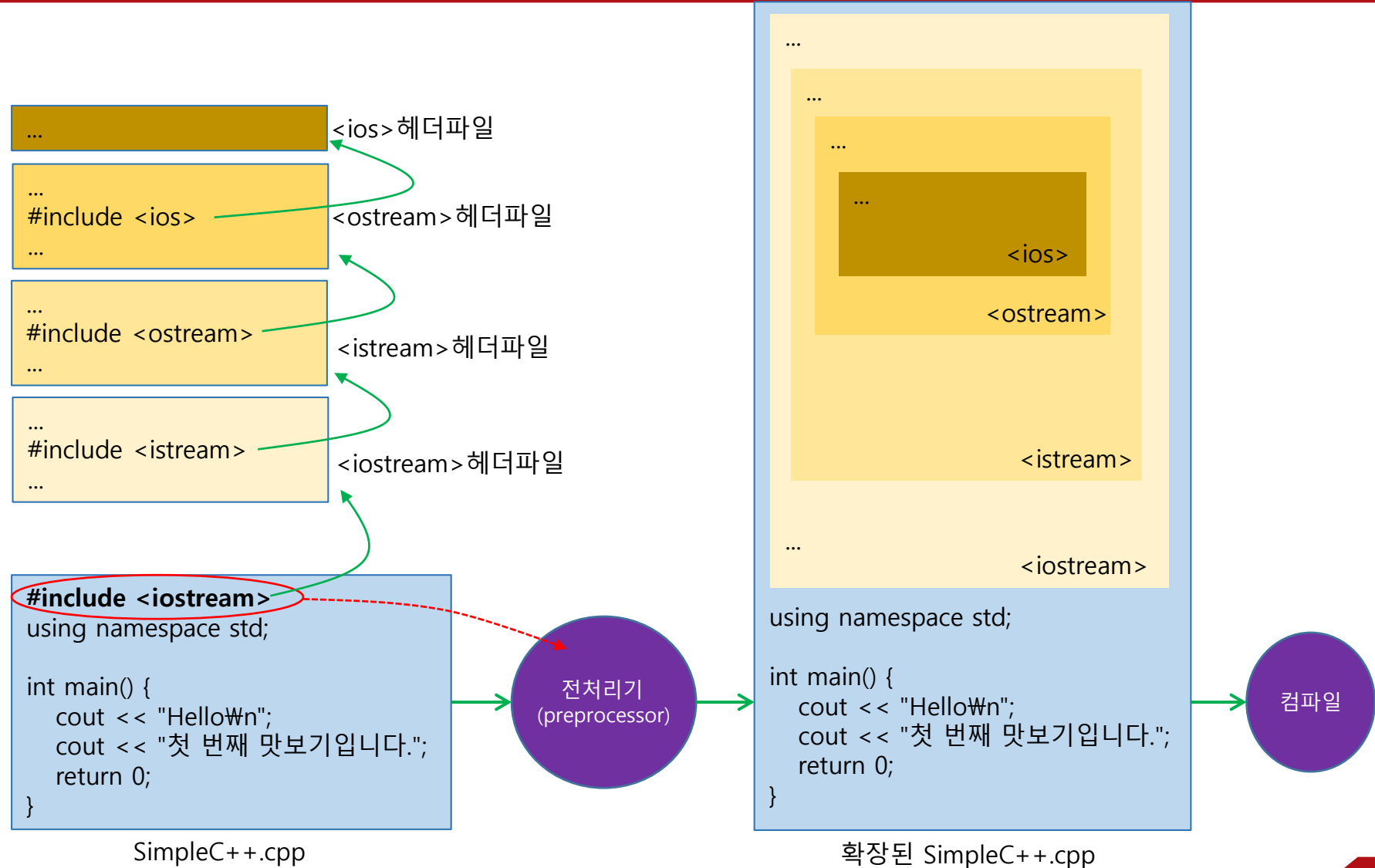
string 클래스를 사용하기 위한 헤더 파일

getline()은 string 타입의
문자열을 입력 받기 위해
제공되는 전역 함수

Falling in love with you를 부른 가수는(힌트 : 첫글자는 E)?Elvis Pride
틀렸습니다. Elvis Presley입니다.

빈칸 포함

#include <iostream>와 전처리기



<iostream> 헤더 파일은 어디에?

- iostream 파일은 확장자 없는 텍스트 파일
- 컴파일러가 설치된 폴더 아래 include 폴더에 존재
 - C:\Program Files(x86)\Microsoft Visual Studio\2017\Community\VC\Tools\MSVC\14.10.25017\include



표준 C++ 헤더 파일은 확장자가 없다

- 표준 C++에서 헤더 파일 확장자 없고, std 이름 공간 적시
 - #include <iostream>
 - using namespace std;
- 헤더 파일의 확장자 비교

언어	헤더 파일 확장자	사례	설명
C	.h	<string.h>	C/C++ 프로그램에서 사용 가능
C++	확장자 없음	<cstring>	using namespace std;와 함께 사용해야 함

#include <헤더파일>와 #include "헤더파일"

- #include <헤더파일>
 - '헤더파일'을 찾는 위치
 - 컴파일러가 설치된 폴더에서 찾으라는 지시
 - 예) #include <iostream>은 iostream 파일을 컴파일러가 설치된 폴더에서 찾으도록 지시
- #include "헤더파일"
 - '헤더파일'을 찾는 위치
 - 개발자의 프로젝트 폴더나
 - 개발자가 컴파일 옵션으로 지정한 include 폴더에서 찾으도록 지시



THANKS FOR YOUR ATTENTION



나사렛대학교
KOREA NAZARENE UNIVERSITY