



고급프로그래밍

C++ 프로그래밍의 기본

정문호 교수
로봇 비전 및 지능 연구실
광운대학교
(02-940-5625, mhjeong@kw.ac.kr)

Schedule



week	Topics		Homework	Quiz
1	과목소개	교과목 소개 (1), C++ 시작 (2)		
2	C++	C++ 프로그래밍의 기본(3), 클래스와 객체	1	1
3		객체생성과 사용, 함수와 참조	2	2
4		복사 생성자와 함수중복, static, friend, 연산자 중복	3	3
5		상속, 가상함수와 추상클래스	4	4
6		템플릿과 STL, 표준 입출력	5	5
7		파일 입출력		
8				
9	C++	예외처리 및 C 사용, 람다식	6	6
10		멀티스레딩	7	7
11		멀티스레딩, 고급문법	8	8
12		고급문법	9	9
13	병렬 프로그래밍	병렬프로그래밍		
14		병렬프로그래밍		
15	기말고사			

오늘의 학습내용

- C++ 프로그래밍의 기본
 - 표준입출력 : cin, cout
 - 문자열 입력 : getline
 - string 클래스

예제 1: 기본적인 C++ 프로그램

```
/*  
    기본적인 C++ 프로그램  
*/  
  
// 예제 1  
#include <iostream>    // cout과 << 연산자 포함  
  
int main()              // C++은 main() 함수에서부터 실행을 시작한다.  
{  
    std::cout << "Hello\n"; // 화면에 Hello를 출력하고 다음 줄로 넘어 감  
  
    return 0;           // main() 함수가 종료하면 프로그램이 종료됨  
}
```

Hello

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 `return`문 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 << "Hello\n";
```

```
std::cout << "첫 번째 맛보기입니다.";
```

화면 출력

■ cout과 << 연산자 이용

```
std::cout << "Hello\n"; // 화면에 Hello를 출력하고 다음 줄로 넘어감  
std::cout << "This is the first example.";
```

■ 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
 - 오른쪽 피연산자를 왼쪽 스트림 객체에 삽입
 - `cout` 객체에 연결된 화면에 출력
- 여러 개의 << 연산자로 여러 값 출력

```
std::cout << "Hello\n" << "This is the first example.";
```

<< 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;
```

- 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

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


예제 2

```
#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
}
```

```
#5.5-3hello1
n + 5 = 8
면적은 28.26
```

namespace 개념

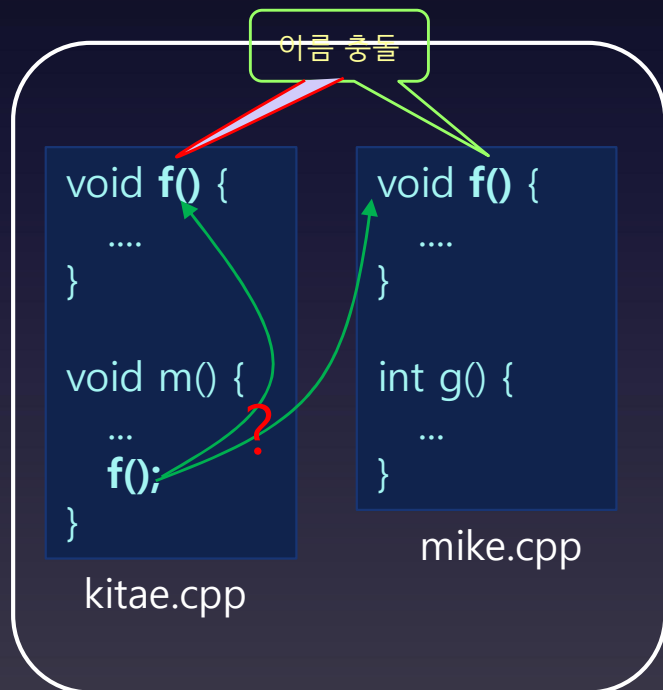
- 이름(identifier) 충돌이 발생하는 경우
 - 여러 명이 서로 나누어 프로젝트를 개발하는 경우
 - 오픈 소스 혹은 다른 사람이 작성한 소스나 목적 파일을 가져와서 컴파일 하거나 링크하는 경우
- 해결하는데 많은 시간과 노력이 필요

- 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

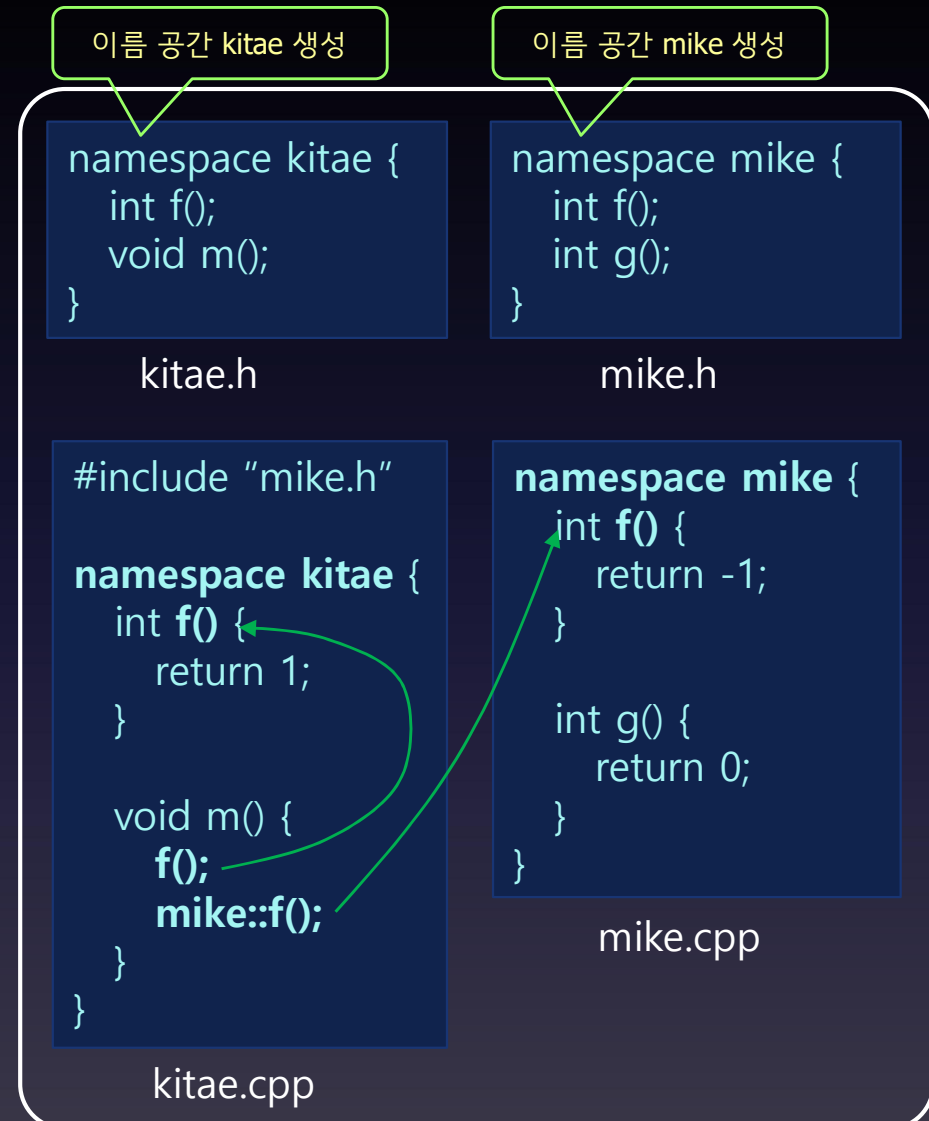
- 이름 공간 생성 및 사용

```
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` 등

■ std:: 생략

- `using` 지시어 사용

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

```
.....
```

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

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

```
.....
```

```
cout << "Hello" << endl; // 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 3

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

int main() {
    cout << "Input a width >> ";

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

    cout << "Input a height >> ";

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

    int area = width*height; // 사각형의 면적 계산
    cout << "The area is " << area << "\n"; // 면적을 출력하고 다음 줄로 넘어감
}
```

```
Input a width >> 3
Input a height >> 5
The are is 15
```

cin과 >> 연산자를 이용한 키 입력

■ cin

- 표준 입력 장치인 키보드를 연결하는 C++ 입력 스트림 객체

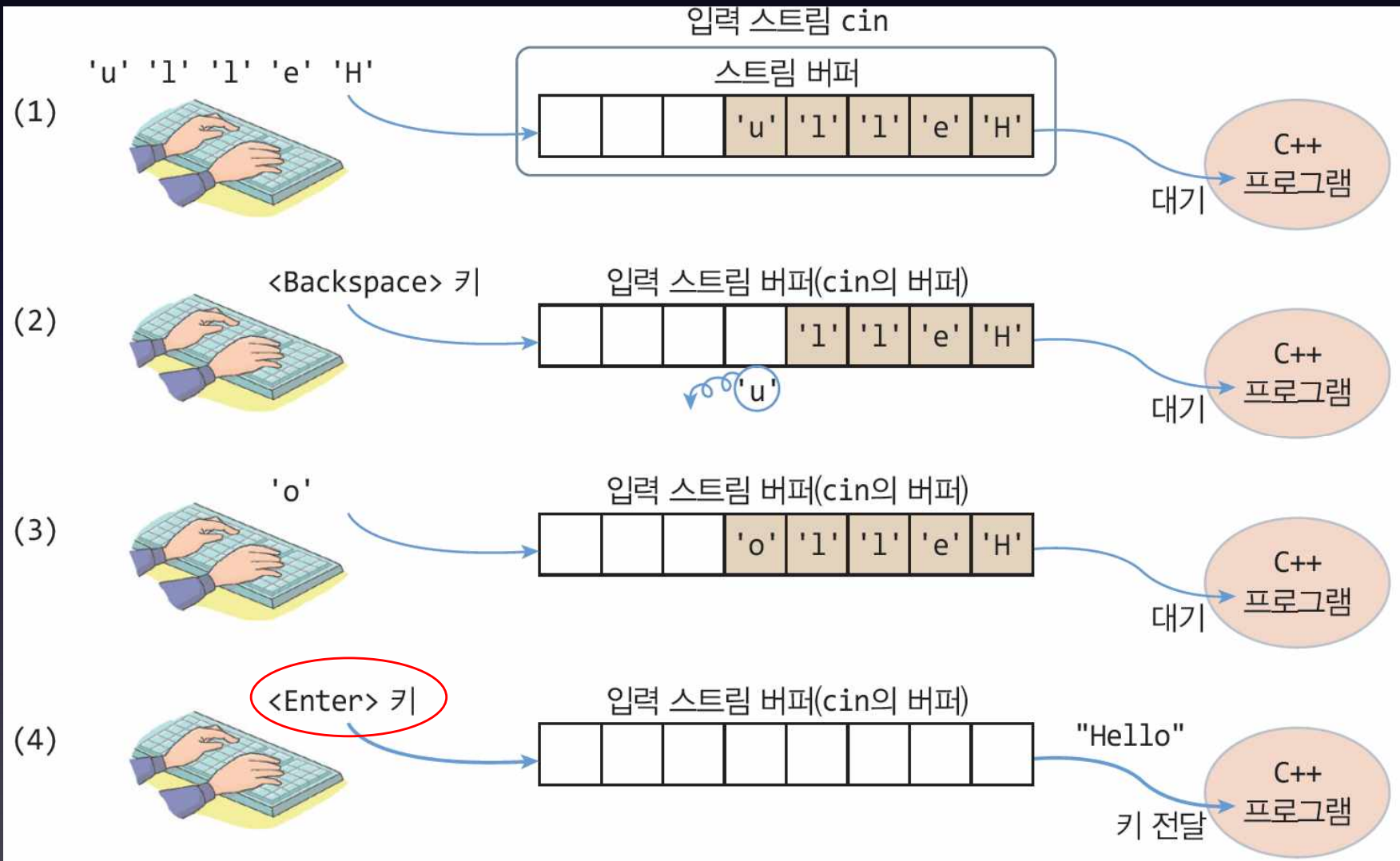
■ >> 연산자

- 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
```

cin으로부터 키 입력 받는 과정



실행문 중간에 변수 선언

■ C++의 변수 선언

- 변수 선언은 아무 곳이나 가능

```
int width;
```

```
cin >> width; // 키보드로부터 너비를 읽는다.
```

```
cout << "Input a height >> ";
```

```
int height;
```

```
cin >> height; // 키보드로부터 높이를 읽는다.
```

```
// 너비와 높이로 구성되는 사각형의 면적을 계산한다.
```

```
int area = width*height;
```

```
cout << "The area is " << area << "\n"; // 면적을 출력하고 한 줄 뺐다.
```

- C++ 변수 선언 방식의 장점

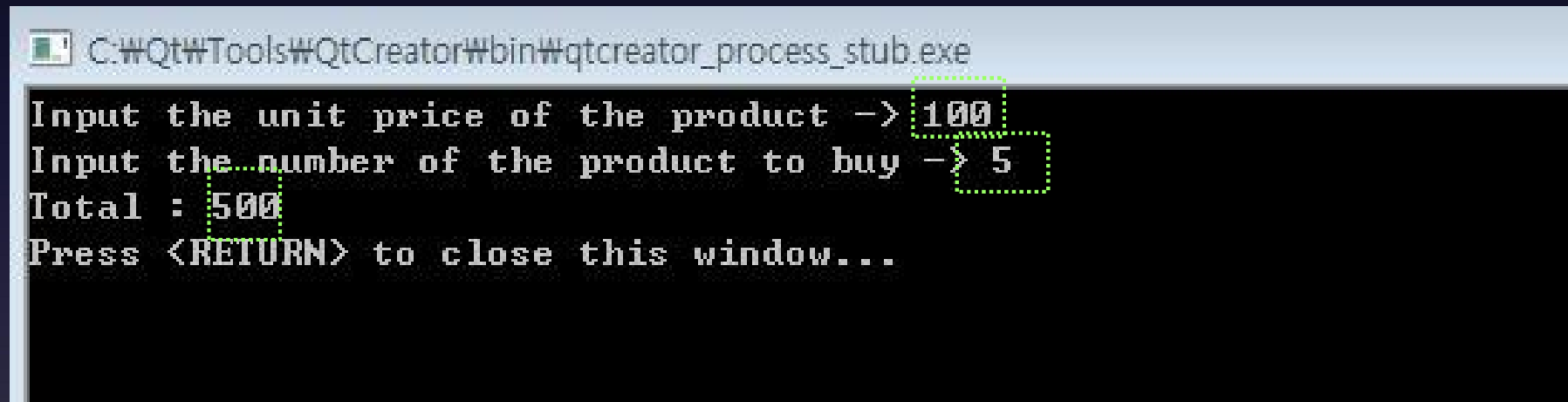
- 변수를 사용하기 직전 선언함으로써 변수 이름에 대한 타이핑 오류 줄임

- C++ 변수 선언 방식의 단점

- 선언된 변수를 일괄적으로 보기 힘들
- 코드 사이에 있는 변수 찾기 어려움

실습 1

- 다음의 결과가 나오도록 `cout`, `cin`을 사용하여 프로그래밍하시오.
 - 부분은 변수로 정의



```
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Input the unit price of the product -> 100
Input the number of the product to buy -> 5
Total : 500
Press <RETURN> to close this window...
```

The screenshot shows a console window titled "C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe". It displays the execution of a program that prompts for user input. The first prompt is "Input the unit price of the product ->" followed by the input "100". The second prompt is "Input the number of the product to buy ->" followed by the input "5". The program then calculates and displays "Total : 500". Finally, it prompts the user to "Press <RETURN> to close this window...". Red dashed boxes highlight the input values "100", "5", and "500" to indicate where variables were defined.

실습 1 - 답

- 다음의 결과가 나오도록 cout, cin을 사용하여 프로그래밍하시오.

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

int main()
{
    int unit, count, total;

    cout << "Input the unit price of the product -> ";
    cin >> unit;

    cout << "Input the number of the product to buy -> ";
    cin >> count;

    total = unit * count;

    cout << "Total : " << total << endl;

    return 0;
}
```

C++ 문자열

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

```
char name1[6] = {'G', 'r', 'a', 'c', 'e', '\0'}; // 문자열
char name2[5] = {'G', 'r', 'a', 'c', 'e'};      // 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-스tring 방식으로 문자열 다루기

- C-스tring으로 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.
```

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

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

"Grace" 문자열

예제 4

```
#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;
    }
}
```

```
프로그램을 종료하려면 암호를 입력하세요.
암호>>Java
암호가 틀립니다~~
암호>>C
암호가 틀립니다~~
암호>>C++
프로그램을 정상 종료합니다.
```

예제 5 — 공백 문자열 입력

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

int main() {
    cout << "Input a name >> ";

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

    cout << "The name is " << name << "\n"; // 이름을 출력한다.
}
```

```
Input a name >> Michael
The name is Michael
```

```
Input a name >> Michael Hong
The name is Michael
```


cin.getline()

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

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

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'	' '	' '	'K'	'o'	'r'	'e'	'a'	'\0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	-----	-----

"Seoul Korea" 문자열

예제 6

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

int main() {
    cout << "Input an address >> ";

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

    cout << "The address is " << address << "\n"; // 주소 출력
}
```

```
Input an address >> Nowon, Seoul
The address is Nowon, Seoul
```

string 클래스

■ string 클래스

- 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 7

```
#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 << "The singer who sang " + song + "is "; // + 로 문자열 연결
    cout << "(hint : The first alphabet is " << elvis[0] << ")?"; // [] 연산자 사용

    getline(cin, singer, '\n'); // 문자열 입력

    if(singer == elvis) // 문자열 비교
        cout << "Correct !";
    else
        cout << "Wrong ! The answer is " + elvis << endl; // +로 문자열 연결
}
```

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

string에 선언된 getline

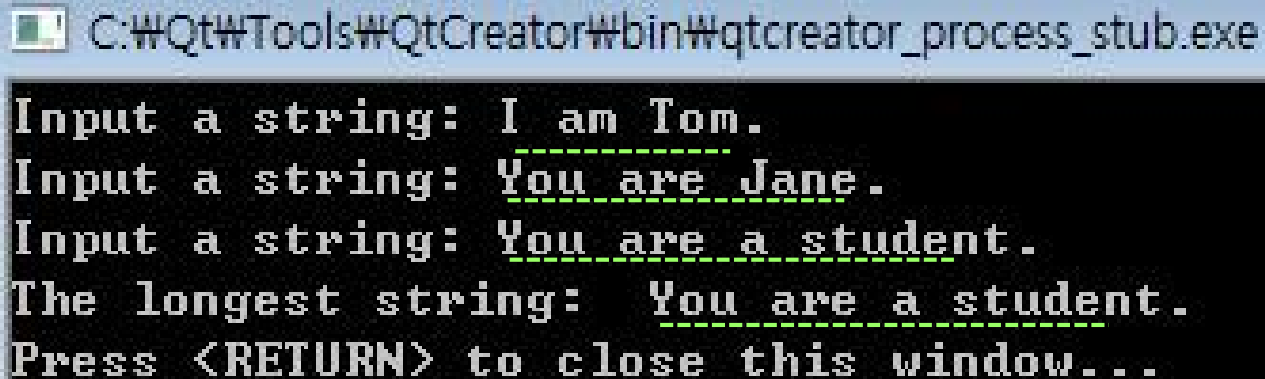
- getline은
 - <iostream>
 - <fstream>
 - <string> 에 정의

```
string str;  
ifstream file("sample.txt");
```

```
getline(cin, str);  
getline(file, str);
```

실습 2

- 다음의 결과가 나오도록 "getline", "cout", "cin"을 사용하여 프로그래밍 하시오.
 - ----- 부분은 변수로 정의
 - 문자열의 길이를 구하는 함수를 사용 : `string::size()`



```
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe  
Input a string: I am Tom.  
Input a string: You are Jane.  
Input a string: You are a student.  
The longest string: You are a student.  
Press <RETURN> to close this window...
```

실습 2 - 답

- 다음의 결과가 나오도록 "getline", "cout", "cin"을 사용하여 프로그래밍하시오.

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

int main()
{
    string a[3];
    for (int i = 0; i<3; i++) {
        cout << "Input a string: ";
        //cin >> a[i];
        getline(cin,a[i]);
    }
    string max=a[0];
    for (int i = 1; i<3; i++) {
        if (a[i].size() > max.size())
            max = a[i];
    }
    cout << "The longest string: " << max << endl;
    return 0;
}
```

