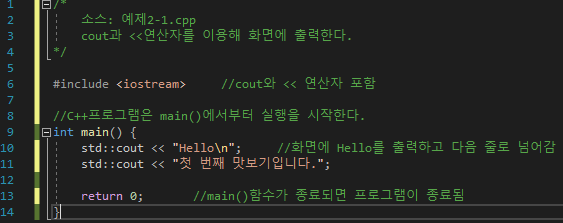
C++프로그래밍의 기본요소와 화면출력

예제2-1 소스



주석문 프로그램의 실행에 영향 X

개발자가 붙인 프로그램의 설명, 메모

여러 줄로 주석 만들기 /\*\*/이용

한 줄로 주석 만들기 //

main()함수 c++프로그램의 실행 시작점

main()함수가 종료되면 프로그램도 종료

리턴타입 C++표준 int

return문 생략가능

#include <iostream> 전처리기 지시문

소스파일을 컴파일하기 전, iostream헤더파일을 읽어 C++소스파일에 삽입

iostream C++표준입출력을 위한 클래스, 객체 선언

화면 출력 cout와 <<연산자 이용

cout 스크린 장치와 연결된 C++표준 출력 스트림 객체

C++프로그램에서 출력한 데이터 🡪 cout 🡪 자신과 연결된 스크린

std:: cout의 네임스페이스가 std임을 표시

<<연산자 스트림삽입 연산자

오른쪽 피연산자 데이터를 왼쪽 스트림 객체에 추가

재정의 기본 정수를 왼쪽으로 시프트하는 C++의 기본 연산자

iostream 출력스트림에 데이터를 삽입하는 삽입연산자로 재정의

한줄에 여러 개의 <<연산자로 한문장에 여러 데이터 출력 가능 왼쪽부터 오른쪽으로 실행

C++기본타입 데이터도 출력 가능 ex) bool, char, short, int, long, float, double

ex) int = 3;

char c = '#';

std::cout << c << 5.5 << '-' << n << "hello" << true;

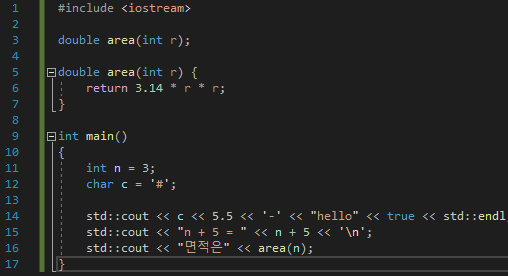
실행결과 #5.5-3hello1

다음줄로 넘어가기 ‘\n’문자사용 ”문자열\n”

endl조작자 사용 std::cout << ”문자열” << std::endl

커서를 다음줄로 넘기고 스트림버퍼에 있는 데이터를 즉각 출력

예제2-2



check time 1. int

2. cout와 <<이 정의되어 있지 않음

3. 1

4. 2

5. std::cout << "서울시 마포구 연남동\n" << "이승재";

namespace와 std::

namespace 이름충돌 이름이 충돌하는 것 ex)함수가 똑같은 게 2개 있으면 어떤 함수를 호출하는지 알 수 없어 오류 발생

서로 다른 네임스페이스에 선언된 이름들은 별개의 이름으로 취급 🡪 충돌 방지

네임스페이스 생성 방법 namespace키워드 이용

namespace name {f(…){…}}

네임스페이스::이름

ex) name::f()

std:: 표준 네임스페이스

모든 C++표준 라이브러리는 std이름공간에 선언 🡪 C++표준라이브러리에서 선언된 이름을 사용할 때 std::를 접두어로 붙여야 함

using 네임스페이스 접두어 생략

ex) using std::cout;

cout << "Hello" << std::endl;

using지시어 아래 모든 코드에서 cout앞에 std::생략가능

ex) std 네임스페이스에 선언된 모든 이름에 대해 std::를 생략

using namespace std;

cout << "Hello" << endl;

check time 1. 4

2. 1

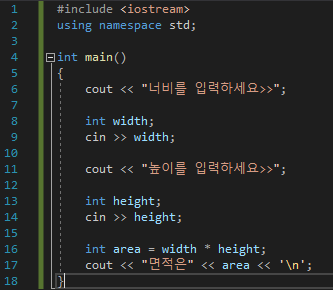
3. std

4. #include <iostream>

using namespace std;

C++프로그래밍의 기본요소와 화면출력

예제2-3 소스



cin과 >>연산자를 이용한 키 입력 <iostream>에 cin과 >>연산자 선언

모든 기본타입에 >>로 입력가능

여러 값을 입력받을 수도 있음 ex)cin >> width >> height;

cin 객체 키보드와 C++응용프로그램을 연결하는 C++표준 입력 스트림 객체

키보드로 입력값은 cin객체의 스트림버퍼로 들어오고 응용프로그램은 cin객체로부터 입력된 값을 받음

키보드 🡪 cin 🡪 C++ 응용프로그램

>>연산자 스트림 추출 연산자

왼쪽 피연산자 스트림객체의 데이터를 오른쪽 피연산자 변수에 삽입

재정의 기본 정수를 오른쪽으로 시프트하는 C++의 기본 연산자

iostream 출력스트림에 데이터를 추출하는 삽입연산자로 재정의

<Enter>키 사용자가 입력한 키들은 cin의 스트림버퍼에 저장

enter키가 입력되면 >>연산자가 cin입력버퍼에서 키값을 끌어내 변수에 저장

실행문 중간에 변수 선언 가능

check time 1. 1

2. 2

3. #include <iostream>

using namespace std;

int main() {

int radius;

cin >> radius;

double area = radius \* radius \* 3.14;

cout << area;

}

키보드로 문자열 입력

C++의 문자열 C-스트링 C언어에서 문자열을 표현하는 방법

string 객체 문자열을 객체로 다루는 방법

C-string 널문자(\0, 0)으로 끝나는 char 배열

c언어에서 사용하는 문자열 방식

char name[6] = { 'g', 'r','a','c','e','\0' }; //문자열

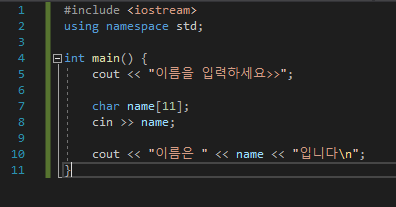
char name[5] = { 'g', 'r','a','c','e'}; //단순 문자 배열

<cstring><string.h> include 🡪 strcpy(), strcmp, strlen()과 같은 C라이브러리 함수 그대로 사용 가능

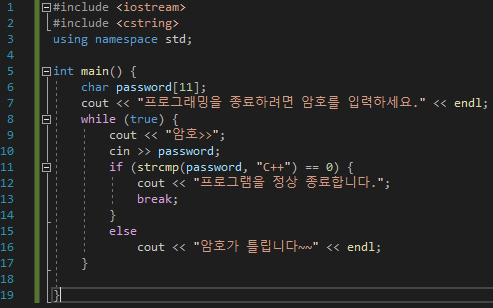
cin을 이용한 문자열 입력 char name[6];

cin >> name

예제2-4 소스



예제2-5 소스



cin과 >>연산자로 문자열을 입력받을 때의 허점 공백문자를 만나면 그 전까지 입력된 문자들을 하나의 문자열로 인식

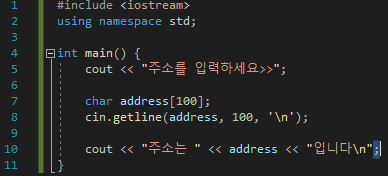
cin.getline()을 이용해 공백이 포함된 문자열 입력 cin.getline(char buf[], int size, char delimitChar) char buf[] 문자열이 저장되는 배열

int size 배열의 크기

char delimitChar 문자열 끝을 지정하는 문자

디폴트 ‘\n’

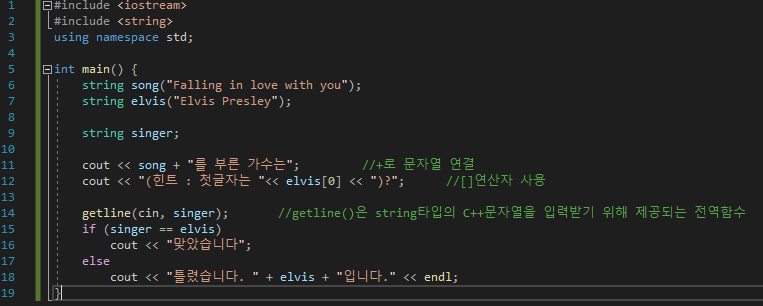
예제2-6



string 클래스 문자열의 크기에 제약 X

문자열을 다루기 쉬움

예제2-7



#include <iostream>

#include <iostream>과 전처리기 컴파일 전 전처리기를 실행해 #include문 모두 처리

헤더파일의 확장 #include문에 지정된 <헤더파일>의 텍스트를 #include문이 있던 자리에 삽입

<iostream>위치 컴파일러가 설치된 폴더의 include 폴더에 존재

헤더파일 확장자 2003년 C++표준부터 헤더파일에 확장자가 없음

#include <헤더파일>과 #include “헤더파일”의 차이 #include <헤더파일> 컴파일러가 설치된 폴더에서 헤더파일을 찾음

#include “헤더파일” 프로젝트폴더, 컴파일옵션으로 지정된 include폴더에서 헤더파일 찾음

헤더파일의 내용 함수원형이 선언되어 있음

ex) strcpy()는 컴파일된 기계어 형태로 C라이브러리에 들어있음

헤더파일은 컴파일할 때 strcpy()의 함수의 호출이 정확한 지 판단

cout, cin의 선언 <iostream>헤더파일에 선언된 객체