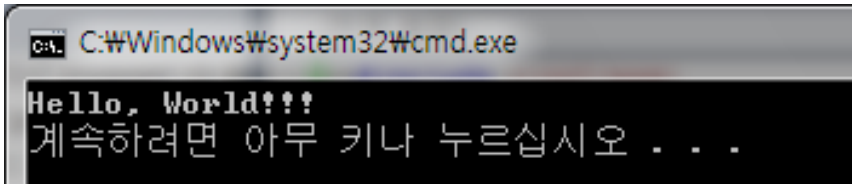
나의 첫 C++ 프로그램

```cpp
#include <iostream>

int main() {
  std::cout << "Hello, World!!!" << std::endl;
  return 0;
}
```

성공적으로 컴파일 하였다면 (이전과 똑같이 `Ctrl + F5` 를 누르면 컴파일 후 빌드까지 하여 프로그램을 출력해줍니다)



우와!!

여러분은 드디어 첫번째 C++ 프로그램을 작성하였습니다. 위 소스가 어떠한 의미를 가지고 있는지는 다음 강좌에서 다루어 보도록 하겠습니다.

첫 C++ 프로그램 분석하기

안녕하세요 여러분. 씹어먹는 C++ 두번째 강좌 입니다. 지난번에는 아마도 여러분 인생 최초의 C++ 프로그램을 만들어 보았을 텐데요, 이번 강좌에서는 소스 코드를 따라가면서 분석을 하는 시간을 갖도록 하겠습니다.

사실, 지금 제 강좌를 보고 계시는 분들 중에서는 막 C 언어 공부를 끝내고 오신 분들도 많으실 텐데요, 무언가 초심자의 마음으로 돌아간 것 같지 않으세요?

C 언어에서 막 어려운 프로그래밍 하다가 C++ 오니 다시 맨 밑바닥 부터 화면에 출력하는 것을 하니 답답한 마음이 들 것도 같네요. 하지만 조금만 기다려보세요. 곧 놀라운 C++ 의 세계가 펼쳐질 것입니다.

```
#include <iostream>

int main() {
  std::cout << "Hello, World!!" << std::endl;
  return 0;
}
```

성공적으로 컴파일 하였다면

실행 결과

```
Hello, World!!
```

와 같이 나옵니다.

위 코드가 바로 지난 강좌에서 사용하였던 코드 입니다. 일단 C 언어와 비슷한 점들 부터 찾아보도록 합시다. 일단 맨 위에

```
#include <iostream>
```

을 보면 아하! *iostream* 이라는 헤더파일을 *include* 하고 있구나 라는 생각이 머리속에 번뜩이셔야 합니다. 그렇지 않다면 C 언어를 다시 공부하도록 하세요! (이 강좌를 보시면 됩니다)

`iostream` 헤더 파일은 C++ 에서 표준 입출력에 필요한 것들을 포함하고 있습니다. 예를 들면 아래에서 사용되는 `std::cout` 이나 `std::endl` 과 같은 것들을 말이지요. C 언어에서의 `stdio.h` 와 비슷하다고 보시면 됩니다. (그리고 C 와 하나 다른 점은 헤더 파일 이름 뒤에 `.h` 가 붙지 않습니다!)

```
int main()
```

네. `main` 함수를 정의하는 부분 입니다. C 와 마찬가지로 C++ 에서의 `main` 함수는 프로그램이 실행될 때 가장 먼저 실행되는 함수 입니다.

그리고 그 함수의 몸체를 보면

```
std::cout << "Hello, World!!" << std::endl;
return 0;
```

와 같은 내용이 있네요. 화면에 대충 출력된 것을 보아 `std::cout` 은 화면에 무언가 출력시켜주는 것 같은데, `printf` 와 다르게 사용된 것을 보니 함수 같지는 않네요. 그리고 화면에 출력된 것을 대충 보면 "계속하려면 아무 키나 누르세요" 가 한 줄 개행되어서 나온 것을 보니 `std::endl` 은 한 줄 엔터를 쳐서 나타내라는 표시 같습니다.

그리고 마찬가지로 `main` 함수에서도 `return` 을 해주고요. 이렇게 대략 살펴보면 기존의 C 언어와 크게 다른 점은 없는 것 같습니다.

하지만 미스테리로 남아있던 부분부터 살펴보도록 합시다.

이름 공간 (namespace)

먼저 `cout` 앞에 붙어 있는 `std` 의 정체부터 알아봅시다. `std` 는 C++ 표준 라이브러리의 모든 함수, 객체 등이 정의된 **이름 공간(namespace)** 입니다.[2)]

그렇다면 이름 공간이란 것이 정확히 무엇일까요? 이름 공간은 말그대로 어떤 정의된 객체에 대해 **어디 소속인지** 지정해주는 것과 동일합니다.

코드의 크기가 늘어남에 따라, 혹은 다른 사람들이 쓴 코드를 가져다 쓰는 경우가 많아지면서 중복된 이름을 가진 함수들이 많아졌습니다. 따라서 C++ 에서는 이를 구분하기 위해, 같은 이름이라도, 소속된 **이름 공간** 이 다르면 다른 것으로 취급하게 되었습니다.

예를 들어서 같은 철수라고 해도, 서울 사는 철수와 부산 사는 철수와 다르듯이 말이지요.

```
std::cout
```

위의 경우 `std` 라는 이름 공간에 정의되어 있는 `cout` 을 의미 합니다. 만약에 `std::` 없이 그냥 `cout` 이라고 한다면 컴파일러가 `cout` 을 찾지 못합니다. 서울에 사는 철수인지 부산에 사는 철수인지 알 길이 없기 때문이지요.

이름 공간을 정의하는 방법은 아래와 같습니다. 예를 들어서 두 헤더파일 `header1.h` 와 `header2.h` 를 생각해봅시다.

2) "표준 라이브러리" 나 "객체" 가 무엇인지 아직 몰라도 괜찮습니다. 그냥 쉽게 생각하자면 stdio.h 가 C 에서 제공하는 라이브러리듯이 iostream 도 C++ 에서 제공하는 출력을 위한 표준 라이브러리 입니다.

```
// header1.h 의 내용
namespace header1 {
int foo();
void bar();
}
```

```
// header2.h 의 내용
namespace header2 {
int foo();
void bar();
}
```

위 코드에서 header1 에 있는 foo 는 header1 라는 이름 공간에 살고 있는 foo 가 되고, header2 에 있는 foo 의 경우 header2 라는 이름 공간에 살고 있는 foo 가 됩니다.

자기 자신이 포함되어 있는 이름 공간 안에서는 굳이 앞에 이름 공간을 명시하지 않고 자유롭게 부를 수 있습니다. 예를 들어서

```
#include "header1.h"

namespace header1 {
int func() {
  foo();  // 알아서 header1::foo() 가 실행된다.
}
}  // namespace header1
```

header1 이름 공간안에서 foo 를 부른다면 알아서 header1::foo() 를 호출하게 됩니다. 그렇다고 해서 header1 의 이름 공간 안에서 header2 의 foo 를 못 호출하는 것은 아닌데 그냥 아래와 같이 간단하게

```
#include "header1.h"

namespace header1 {
int func() {
  foo();           // 알아서 header1::foo() 가 실행된다.
  header2::foo();  // header2::foo() 가 실행된다.
}
}  // namespace header1
```

반면에 어떠한 이름 공간에도 소속되지 않는 경우라면 아래와 같이 명시적으로 이름 공간을 지정해야 합니다.

```
#include "header1.h"
#include "header2.h"
```

```
int func() {
  header1::foo();  // header1 이란 이름 공간에 있는 foo 를 호출
}
```

하지만 만일 위 같은 foo 을 여러번 반복적으로 호출하게 되는 경우 앞에 매번 header1:: 을 붙이기가 상당히 귀찮을 것입니다.

그래서 아래와 같이 '나는 앞으로 header1 이란 이름 공간에 들어있는 foo 만 쓸거다!' 라고 선언할 수 있습니다.

```
#include "header1.h"
#include "header2.h"

using header1::foo;
int main() {
  foo();  // header1 에 있는 함수를 호출
}
```

뿐만 아니라, 그냥 기본적으로 header1 이름 공간안에 정의된 모든 것들을 header1:: 없이 사용하고 싶다면

```
#include "header1.h"
#include "header2.h"

using namespace header1;
int main() {
  foo();  // header1 에 있는 함수를 호출
  bar();  // header1 에 있는 함수를 호출
}
```

아예 위와 같이 using namespace header1 과 같이 명시하면 됩니다.

물론 이 경우 역시 header2 에 있는 함수를 못 사용하는 것은 아니고 다음과 같이 명시적으로 써주면 됩니다.

```
#include "header1.h"
#include "header2.h"
using namespace header1;

int main() {
  header2::foo();  // header2 에 있는 함수를 호출
  foo();           // header1 에 있는 함수를 호출
}
```

그렇다면 다시 원래 예제를 살펴보도록 합시다.

```
int main() {
  std::cout << "Hello, World!!" << std::endl;
  return 0;
}
```

여기서 cout 과 endl 은 모두 iostream 헤더파일의 std 라는 이름 공간에 정의되어 있는 것들입니다. std 를 붙이기 귀찮은 사람의 경우에는 그냥

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

int main() {
  cout << "Hello, World!!" << endl;
  return 0;
}
```

로 써도 됩니다.

주의 사항

참고로 `using namespace std;` 와 같이 어떠한 이름 공간을 사용하겠다라고 선언하는 것은 권장되지 않습니다. 왜냐하면 std 에 이름이 겹치는 함수를 만들게 된다면, 오류가 발생하기 때문이지요.
게다가 C++ 표준 라이브러리는 매우 매우 거대하므로, 정말 수 많은 함수들이 존재하고 있습니다. 자칫 잘못하다가 이름을 겹치게 사용한다면, 고치느라 시간을 많이 잡아먹을 것입니다.
게다가 std 에는 매번 수 많은 함수들이 새롭게 추가되고 있기 때문에 C++ 버전이 바뀔 때 마다 기존에 잘 작동하던 코드가 이름 충돌로 인해 동작하지 않게되는 문제가 발생할 수 있습니다.
따라서 **권장하는 방식은 `using namespace std;` 같은 것은 사용하지 않고, `std::` 를 직접 앞에 붙여서 std 의 이름공간의 함수이다 라고 명시해주는 것이 좋습니다.** 또한, **여러분이 작성하는 코드는 여러분 만의 이름 공간에 넣어서 혹시 모를 이름 충돌로 부터 보호하는 것이 중요합니다.**

그렇다면 cout 은 무엇일까요? 정확히 무엇인지 말하자면 ostream 클래스의 객체로 표준 출력(C 언어에서의 stdout 에 대응됩니다) 을 담당하고 있습니다.

무슨 말인지 모르겠다고요? 괜찮습니다. 이 것이 정확히 무슨 의미인지는 나중 강좌에서 알아보도록 하겠고, 그냥 다음과 같이 쓴다는 것만 알아두시면 됩니다.

```
std::cout << /* 출력할 것 */ << /* 출력할 것 */ << ... << /* 출력할 것 */;
```

그리고 endl 은 화면에 출력해주는 '함수' 입니다. 놀라셨지요? 하지만 그냥

```
std::cout << std::endl;
```

이라 쓰면 화면에 엔터를 하나 출력해주는 것으로 기억하시면 됩니다. 물론 `endl` 에 대해서도 나중에 다루어 보도록 하겠습니다 :)

이름 없는 이름 공간

잠깐 짚고 넘어가자면, C++ 에서는 재미있게도 이름 공간에 굳이 이름을 설정하지 않아도 됩니다.

이 경우 해당 이름 공간에 정의된 것들은 해당 파일 안에서만 접근할 수 있게 됩니다. 이 경우 마치 `static` 키워드를 사용한 것과 같은 효과를 냅니다.

```
#include <iostream>

namespace {
// 이 함수는 이 파일 안에서만 사용할 수 있습니다.
// 이는 마치 static int OnlyInThisFile() 과 동일합니다.
int OnlyInThisFile() {}

// 이 변수 역시 static int x 와 동일합니다.
int only_in_this_file = 0;
}  // namespace

int main() {
  OnlyInThisFile();
  only_in_this_file = 3;
}
```

예를 들어서 위 경우 `OnlyInThisFile` 함수나 `only_in_this_file` 변수는 해당 파일 안에서만 접근할 수 있습니다. 헤더파일을 통해서 위 파일을 받았다 하더라도 (물론 `main` 함수 부분은 무시하고), 저 익명의 `namespace` 안에 정의된 모든 것들은 사용할 수 없게 됩니다.

생각 해보기

문제 1

화면에 출력되는 것들을 바꾸어보자.

문제 2

아래 문장은 화면에 어떻게 출력될까요?

```
std::cout << "hi" << std::endl
          << "my name is "
          << "Psi" << std::endl;
```