# Chap2

main() 함수
- 리턴 타입으로 void를 쓰기도 하지만 표준: int
- return문은 생략이 가능

#include <iostream>
- c++의 표준 입출력 전처리기
  (표준 입출력에 필요한 클래스, 객체 등 선언)
  ex) cout객체, cin객체
- 키보드 입력, 화면 출력에 필요

화면 출력

cout 객체
- c++ 표준 출력스트림객체

<<연산자
- 스트림 삽입 연산자
- 오른쪽 객체를 왼쪽 출력 스트림 객체에 삽입
- 기본 산술 시프트 연산자가 재정의 된 것

```
std::cout << "hi\n";  -> hi(줄바꿈)
std::cout <<"Hello"<<endl;  -> Hello(줄바꿈)
std::cout << f(); -> 함수의 리턴값 출력
```

endl
- endl() 함수를 호출하여 cout에게 현재 버퍼에 있는 데이터를 즉각 출력
- iostream에 정의

namespace(이름공간) 키워드
- 공동 개발이나 다른 소스 코드등을 가져와 작업할 경우 같은 이름인데 기능이나 내용이 다른 경우가 발생해 이름 충돌이 일어날 수 있다
- 따라서 namespace 키워드를 사용해 이름공간에 변수,함수,클래스 등의 이름을 적는다

* c++ 표준 라이브러리는 모두 std에 선언되어 있기 때문에 c++표준 헤더파일의 클래스,함수,객체들도 std에 선언되어 있다

```
namespace Kitae{
  int f();
  void m();
}
```
kitae.h

```
namespace mike{
  int f();
  int g();
}
```
mike.h

```
#include "mike.h"
namespace kitae{
  int f(){
    return 1;
  }
  int m(){
    f();
    mike::f();
  }
}
```
kitae.cpp

```
namespace mike{
  int f(){
    return -1;
  }
  int g(){
    return 0;
  }
}
```
mike.cpp

헤더 파일에 namespace를 생성하고
소스파일에서 정의들을 작성한다. 드럽게 복잡하네,,,

다른 namespace를 가져올 땐 include를 통한다
또한 반드시 :: 기호를 써줘야 한다.

using 지시어
- using namespace std
  => std에 있는 모든 이름 std:: 없이 사용 가능
- using std::cout
  => cout만 std:: 없이 사용 가능

* iostream 헤더파일에 std 이름공간 존재

입력

cin 객체
- c++ 표준 입력스트림객체
- 키보드로부터 입력되는 값들을 모두 cin 객체 스트림 버퍼로 들어옴

>> 연산자
- 스트림 추출 연산자( << 처럼 재정의 된 것)

* <enter> 를 치면 cin 스트림버퍼의 값들이 변수로 저장