CHAPTER 02

C++ 함수 & 네임스페이스

디폴트 매개변수

- 매개변수의 디폴트 값을 선언한 함수는 호출자 코드에서 실인수를 생략한 채 호출할 수 있다.

```
int Test(int n=10) {
    return n;
}

int main() {
    cout << Test() << '\n';
    cout << Test(10) << '\n';
    return 0;
}
```

```
10
10
```

- 호출자 함수가 피호출자 함수 매개변수의 실인수를 기술하면 왼쪽부터 짝을 맞추어 적용되며, 짝이 맞지 않는 매개변수는 디폴트 값 적용

- 디폴트 값이 없는 매개변수에는 호출자 함수에 반드시 실인수 기술

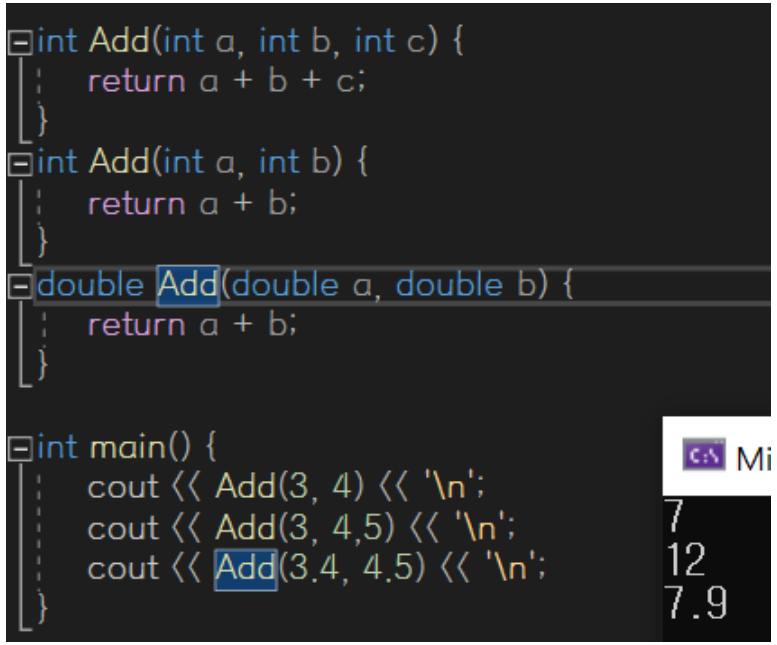
- 중간에 위치한 매개변수에 디폴트 값 생략할 수 없다.

```
int Test(int n=10, int n2, int n3=20) {
    return n;
}
```

함수 다중 정의

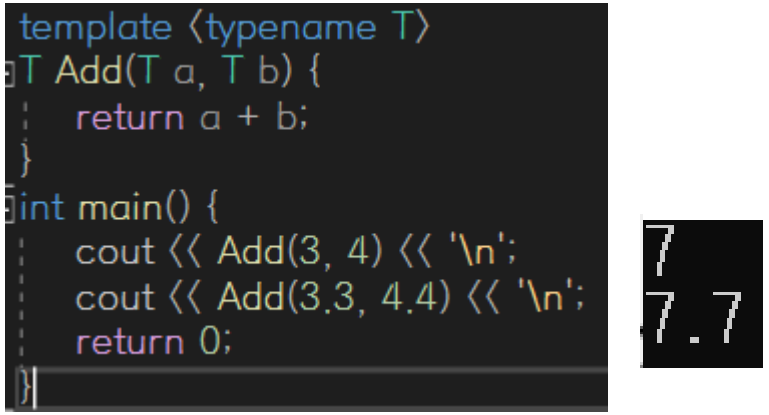
다중정의

(같은 일을 하는 함수가 여러 개 존재하지만 실제 사용되는 것이 하나 밖에 없어 불필요한 코드 늘어나 메모리 소모 큼)



함수 템플릿

일종의 틀, 여러 장의 판화 인쇄하는 것
같은 일을 하는 코드가 여러 번 등장할 필요가 없어졌기 때문에 안정적인 구조



인라인 함수

함수 호출 -〉 스택 메모리 사용 증가, 매개변수로 인해 메모리 복사 발생 & 제어 흐름 이동

매크로 -〉 관리상의 목적. 본질은 함수가 아니므로 오류 발생시키기도 함. 매개변수 형식 지정 할 수 없음

➔인라인 함수 : 함수와 매크로 장점만 모아 놓은 것

visual studio 기본값 설정 : 인라인 함수로 사용하는 데 적합한 함수는 모두 인라인 함수로 만든다. 따라서 VS 이용해 컴파일 할 경우 인라인으로 선언하지 않았다고 하더라도 인라인 함수로 확장. 또한 인라인으로 선언했다고 해도 적합하지 않다면 일반 함수로 컴파일함.

네임스페이스

C++ 가 지원하는 각종 요소들을 한 범주로 묶어주기 위한 문법.

소속, 구역이라는 개념.

문법에 따라 임의로 생략하는데, 예약어가 using

⇒using namespace 네임스페이스이름;

네임스페이스 중첩 : 네임스페이스 안에 또 다른 네임스페이스 속할 수 있다. (ex. 서울시::강남구::대치동::xx로::xx길)

식별자 검색 순서

- 전역 함수인 경우

현재 블록 범위 -> 상위 블록 범위 -> 최근 선언된 전역 변수나 함수 -> using 선언된 네임스페이스 혹은 전역 네임스페이스

- 클래스 메서드인 경우

현재 블록 범위 -> 상위 블록 범위 -> 클래스 멤버 -> 부모 클래스 멤버 -> 최근 선언된 전역 변수나 함수 -> 호출자 코드 속한 네임스페이스의 상위 네임스페이스 -> using 선언된 네임스페이스 혹은 전역 네임스페이스