

매개변수의 '디폴트 값(기본 인수)' 이해하기 🧠

앞서 함수 오버로딩을 봤듯이, C++의 함수 매개변수에는 디폴트 값(기본 인수, **default argument**) 을 지정할 수 있다. 여기서 말하는 '디폴트 값'이란 호출 시 인자를 생략했을 때 대신 사용되는 값을 의미한다.

1) 기본 예시

```
int MyFuncOne(int num = 7)
{
    return num + 1;
}

int MyFuncTwo(int num1 = 5, int num2 = 7)
{
    return num1 + num2;
}
```

- MyFuncOne의 선언 `int num = 7` 은
👉 "인자를 전달하지 않으면 7이 전달된 것으로 간주하겠다"는 뜻이다.

```
MyFuncOne();    // 7이 전달된 것으로 간주
MyFuncOne(7);  // ↑ 위 호출과 동일
```

- MyFuncTwo의 선언 `int num1 = 5, int num2 = 7` 은
👉 "인자를 생략하면 num1에는 5, num2에는 7을 넣겠다"는 뜻이다.

```
MyFuncTwo();    // 5, 7이 전달된 것으로 간주
MyFuncTwo(5, 7); // ↑ 위 호출과 동일
```


2) 예제: DefaultValue1.cpp

```
#include <iostream>

int Adder(int num1 = 1, int num2 = 2)
{
    return num1 + num2;
}

int main()
{
    std::cout << Adder() << std::endl;    // 1 + 2
    std::cout << Adder(5) << std::endl;   // 5 + 2
}
```

```
std::cout << Adder(3, 5) << std::endl;    // 3 + 5
}
```

여기서 알 수 있는 사실 

- 선언된 매개변수 수보다 적은 수의 인자 전달이 가능하다.
- 전달하는 인자는 왼쪽부터 순서대로 채워지고, 부족한 부분은 디폴트 값으로 채워진다.

디폴트 값은 '선언부'에만 적는다

함수 원형을 별도로 선언하는 경우, 디폴트 값은 함수의 선언부(원형)에만 뒤야 한다.


3) 예제: DefaultValue2.cpp

```
#include <iostream>
int Adder(int num1 = 1, int num2 = 2); // ← 선언(원형) + 디폴트 값

int main()
{
    std::cout << Adder() << std::endl;
    std::cout << Adder(5) << std::endl;
    std::cout << Adder(3, 5) << std::endl;
}

int Adder(int num1, int num2) // ← 정의(구현). 여기엔 디폴트 값 X
{
    return num1 + num2;
}
```

왜 선언부에만?

- 디폴트 인수는 호출 지점에서 해석된다. 즉, 호출하는 코드가 보는 선언에 있어야 한다.
- 또한 한 매개변수에 디폴트 인수를 한 번만 줄 수 있다. 선언부·정의부에 중복으로 쓰면 에러! 

```
int f(int x = 1);
int f(int x = 2); //  error: default argument given more than once
```

부분적 디폴트 값 설정

다음처럼 전부 줄 수도 있고,

```
int YourFunc(int num1 = 3, int num2 = 5, int num3 = 7) { /* ... */ }
```

일부만 줄 수도 있다.

```
int YourFunc(int num1, int num2 = 5, int num3 = 7) { /* ... */ }
```

이 경우 호출은 이렇게 동작한다:

```
YourFunc(10);           // YourFunc(10, 5, 7)
YourFunc(10, 20);       // YourFunc(10, 20, 7)
```

하지만 아래처럼 오른쪽 매개변수를 비워두고 왼쪽만 디폴트 지정하는 건 불가다 ❌

```
int YourFunc(int num1 = 3, int num2 = 5, int num3); // (X)
```

즉, 오른쪽부터 연속해서 디폴트를 채워야 한다.

```
int YourFunc(int num1, int num2, int num3 = 30);           // (O)
int YourFunc(int num1, int num2 = 20, int num3 = 30);      // (O)
int YourFunc(int num1 = 10, int num2 = 20, int num3 = 30); // (O)
```

아래 둘은 전부 불가:

```
int WrongFunc(int num1 = 10, int num2, int num3);           // (X)
int WrongFunc(int num1 = 10, int num2 = 20, int num3);      // (X)
```

이유는?

인자는 원→오로 채워지므로, 왼쪽에 디폴트가 있고 오른쪽에 값이 비어 있으면 호출 시 모순이 생긴다.

그래서 오른쪽부터 디폴트를 연속으로 두는 형태만 의미가 있다. 💡

종합 예제: DefaultValue3.cpp

라벨과 호출 인자를 일치시키도록 표기도 살짝 교정.

```
#include <iostream>
int BoxVolume(int length, int width = 1, int height = 1);

int main()
{
    std::cout << "[3, 3, 3] : " << BoxVolume(3, 3, 3) << std::endl;
```

```

std::cout << "[5, 3, D] : " << BoxVolume(5, 3) << std::endl; // ← 라벨
교정
std::cout << "[7, D, D] : " << BoxVolume(7) << std::endl;
// std::cout << "[D, D, D] : " << BoxVolume() << std::endl; // length엔
디폴트가 없으므로 불가

return 0;
}

int BoxVolume(int length, int width, int height)
{
    return length * width * height;
}

```

참고: `length`까지도 생략 호출을 허용하고 싶다면 선언을

`int BoxVolume(int length = 1, int width = 1, int height = 1);` 처럼 바꿔야 한다.

한눈에 정리

- 디폴트 값 = 생략된 인자를 대신할 값
- 인자 전달은 원→오로 채워지고, 부족분은 디폴트로 보충
- 선언부(원형)에만 디폴트 값을 적고, 중복 지정 금지
- 오른쪽부터 연속해서 디폴트 지정