⚙️ 함수 오버로딩의 이해

❌ C 언어에서는 **같은 이름**의 함수를 중복 정의할 수 없다(컴파일 오류).
✅ C++에서는 **매개변수 목록이 다르면** 같은 이름의 함수를 **오버로딩**해서 정의할 수 있다.

- ✅ C++ 허용 예(이름 동일, 매개변수 시그니처 다름):

```
int MyFunc(int num) {
    num++;
    return num;
}

int MyFunc(int a, int b) {
    return a + b;
}

int main(void) {
    MyFunc(20);      // MyFunc(int) 호출
    MyFunc(30, 40);  // MyFunc(int, int) 호출
    return 0;
}
```

- 💡 왜 C++은 되고 C는 안 될까?
  - **C++**: 호출 대상을 찾을 때 **함수 이름 + 매개변수 선언(타입/개수)** 을 함께 본다 → 오버로딩 가능
  - **C**: 호출 대상을 찾을 때 **함수 이름만** 본다 → 오버로딩 불가(문법적으로도 허용하지 않음)

🧩 함수 오버로딩의 예

함수 오버로딩이 가능하려면 **매개변수 선언이 달라야** 한다. 예를 들어, 다음 두 함수는 오버로딩이 가능하다.

```
int MyFunc(char c) { /* ... */ }
int MyFunc(int n)  { /* ... */ }
```

- 자료형이 다르므로, **전달 인자의 타입**으로 어떤 함수를 호출할지 구분할 수 있다. ✅

마찬가지로 다음 두 함수도 오버로딩이 가능하다.

```
int MyFunc(int n)          { /* ... */ }
int MyFunc(int n1, int n2) { /* ... */ }
```

- 매개변수 **개수**가 다르므로, **인자 개수**로 호출 대상을 구분한다. ✅

정리하면, 오버로딩은 **매개변수의 자료형 또는 개수가 달라야** 한다. ✔

반면, 다음은 **잘못된** 오버로딩이다(반환형만 다름).

```
void MyFunc(int n) { /* ... */ }
int  MyFunc(int n) { /* ... */ }
```

- 반환형은 **구분 기준이 아니다** → 컴파일 오류. ❌

FunctionOverloading.cpp

```cpp
#include <iostream>

void MyFunc(void)
{
    std:: cout << "MyFunc(void) called"<< std::endl;
}

void MyFunc(char c)
{
    std::cout << "MyFunc(char c) called" << std::endl;
}

void MyFunc(int a, int b)
{
    std::cout << "MyFunc(int a, int b) called" << std::endl;
}

int main(void)
{
    MyFunc();
    MyFunc ('A');
    MyFunc(12, 13) ;

    return 0;
}
```

이렇듯 오버로딩은 어렵지 않다. **이름은 같고, '입력(매개변수)'만 다르면** 다른 함수로 인식하는 **자연스러운 개념**이다. 💫