🖨️ printf와 scanf를 대신하는 입출력 방식

C++도 보통 "Hello World" 예제로 시작한다. "Hello World"를 출력하는 예제를 실행해 보자.

> 1_Hello_World.cpp

```cpp
#include <iostream>

int main()
{
    int num = 20;
    std::cout << "Hello World!" << std::endl;
    std::cout << "Hello " << "World!" << std::endl;
    std::cout << num << ' ' << 'A';
    std::cout << ' ' << 3.14 << std::endl;

    return 0;
}
```

🔍 관찰결과 1 : 헤더 파일 선언문 `#include <iostream>`

C 언어에서는 입출력을 위해 `printf`, `scanf`를 사용할 때 헤더 파일 `<stdio.h>`를 포함한다.

C++에서는 표준 입출력을 위해 다음 선언이 필요하다.

```cpp
#include <iostream>
```

이 선언이 없으면 다음과 같은 오류가 난다. "std, cout, endl... 이게 다 뭡니까?"

즉, `std`, `cout`, `endl`을 쓰려면 위 헤더 선언이 반드시 필요하다.

C++ 표준 헤더는 확장자 `.h`를 붙이지 않는다.
프로그래머가 직접 만든 헤더는 `.h`를 쓰지만, 표준 라이브러리 헤더는 확장자를 생략한다.

과거:

```cpp
#include <iostream.h> // 구 표준 라이브러리
```

현재:

```cpp
#include <iostream> // 신 표준 라이브러리
```

확장자 생략 이유

1. 구/신 표준 라이브러리를 구분하기 위해
2. 코드를 신 표준 형태로 쉽게 전환하기 위해

참고

- `<iostream.h>` : 구 표준 입출력 라이브러리
- `<iostream>` : 신 표준 입출력 라이브러리
- 최신 C++ 컴파일러는 `<iostream.h>` 지원을 점차 중단

🔍 관찰결과 2 : `std::cout`과 `<<` 연산자를 이용한 출력

기본 형식

```
std::cout << 출력대상;
```

출력대상에는 정수, 실수, 문자열, 변수 등 무엇이든 올 수 있다.
C의 `printf`처럼 `%d`, `%s` 같은 서식 문자를 쓰지 않아도 자료형에 맞춰 자동으로 출력된다.
실무에서도 간단하고 읽기 쉬워서 많이 쓴다.

🔍 관찰결과 3 : `<<` 연산자의 연쇄 출력과 개행

`<<`는 **연산자**이며, 이를 이어 붙여 여러 값을 한 줄로 출력할 수 있다.

예시

```
std::cout << "Hello " << "World!" << std::endl;
std::cout << num << ' ' << 'A';
std::cout << ' ' << 3.14 << std::endl;
```

마지막 예시의 의미

- 공백 문자 출력 → 3.14 출력 → `std::endl`로 줄바꿈

`std::endl`은 개행을 의미한다.
이를 지우고 실행하면 줄바꿈이 사라진다.

여기까지가 기본 출력 방법이다.
`std::cout`, `<<`, `std::endl`의 내부 동작은 추후에 더 다룬다.

⌨️ scanf를 대신하는 데이터 입력

- 입력을 쓰려면 `#include <iostream>`가 필요하다.
- 입력에는 `std::cin`과 `>>` 연산자를 사용한다.
- 변수는 함수 안 **어디서든** 선언할 수 있다.

> 2_SimpleAdder.cpp

```cpp
#include <iostream>

int main()
{
    int val1;
    std::cout << "첫 번째 숫자 입력: ";
    std::cin >> val1;

    int val2;
    std::cout << "두 번째 숫자 입력: ";
    std::cin >> val2;

    int result = val1 + val2;
    std::cout << "덧셈 결과: " << result << std::endl;

    return 0;
}
```

🔍 관찰결과 1 : 데이터 입력(`std::cin`)

- 기본 형식

```
std::cin >> 변수;
```

  - 변수 위치에는 입력받은 값을 저장할 변수명이 온다.
- 예

```
std::cin >> val1; // 정수를 입력받아 val1에 저장
```

- 실수 입력
  - 변수를 `double`로 선언하면 포맷 지정 없이 실수 입력 가능
  - `int` 변수엔 정수, `double` 변수엔 실수가 자동으로 들어간다.
- 문자열 입력
  - C

```c
char str[100];
scanf("%s", str);
```

  - C++

```cpp
char str[100];
std::cin >> str;
```

🔍 관찰결과 2 : C++의 지역변수 선언

- C++에서는 함수 안 **어디서든** 지역 변수를 선언할 수 있다.
- 대부분의 컴파일러가 변수 선언 위치에 제한을 두지 않는다.
- `for` 초기화 구문에서도 변수 선언 가능

```
for (int num = 0; num < 10; num++) { ... }
```

- 연속 입력 형식

```
std::cin >> 변수1 >> 변수2;
```

  - 첫 입력 → 변수1
  - 두 번째 입력 → 변수2
  - 값 구분은 공백(스페이스, 탭, Enter)로 처리된다.

> 3_BetweenAdder.cpp

```cpp
#include <iostream>

int main()
{
    int val1, val2;
    int result = 0;

    std::cout << "두 개의 숫자 입력: ";
    std::cin >> val1 >> val2;

    if (val1 < val2)
    {
        for (int i = val1 + 1; i < val2; i++)
            result += i;
    }
    else
    {
        for (int i = val2 + 1; i < val1; i++)
            result += i;
    }

    std::cout << "두 수 사이의 정수 합: " << result << std::endl;

    return 0;
}
```

📝 배열 기반의 문자열 입출력

> 4_StringIO.cpp

```cpp
#include <iostream>

int main()
{
    char name[100];
    char lang[200];

    std::cout << "이름은 무엇입니까? ";
    std::cin >> name;

    std::cout << "좋아하는 프로그래밍 언어는 무엇인가요? ";
    std::cin >> lang;

    std::cout << "내 이름은 " << name << "입니다.\n";
    std::cout << "제일 좋아하는 언어는 " << lang << "입니다." << std::endl;

    return 0;
}
```

- 구조는 앞선 예제와 같고, 대상만 **문자열**이다.
- C++에서도 배열 기반의 문자열 입출력이 가능하다.