🖨️ printf와 scanf를 대신하는 입출력 방식

C++도 "Hello world" 출력 예제로 시작한다.
C 버전과 조금 다르지만, 초보자는 먼저 눈에 익히고 외우는 과정이 정상적이다.

"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` 함수를 호출할 때
헤더 파일 `<stdio.h>`를 포함한다.
C++에서는 표준 입출력을 위해 다음 선언을 추가해야 한다.

#include

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

즉, `std`, `cout`, `endl`을 사용하려면 위의 헤더 파일 선언문이 필요하다.

C++ 표준 헤더 파일 선언에서는 확장자 `.h`를 생략한다.
프로그래머가 정의하는 헤더 파일은 `.h`를 사용하지만,
표준 라이브러리는 `.h`를 붙이지 않는다.

과거: #include <iostream.h> // 구 표준 라이브러리

현재: #include // 신 표준 라이브러리

확장자 생략 이유:

1. 구 표준 라이브러리와 신 표준 라이브러리를 구분하기 위해
2. 소스 코드를 쉽게 신 표준 형태로 변경할 수 있도록 하기 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

출력을 위해서는 다음과 같이 작성한다. std::cout << 출력대상;

출력대상에는 정수, 실수, 문자열, 변수 등이 올 수 있다.
C 언어의 `printf`처럼 `%d`, `%s`와 같은 서식 문자를 사용하지 않아도
데이터의 자료형에 맞게 자동으로 출력된다.
이 방식은 C 언어의 출력 방식보다 편리하다.

🔍 관찰결과 3 : `<<` 연산자를 이용한 출력 대상의 연이은 표현과 개행

`<<`는 연산자이며, 이를 이용하면 둘 이상의 출력을 연이어 작성할 수 있다.

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

마지막 예시는 다음을 의미한다.

- 공백 문자 출력 → 3.14 출력 → `std::endl` 출력

`std::endl`은 개행을 의미한다.
예제에서 `std::endl`을 제거하고 실행하면 개행이 사라진다.

이로써 기본적인 데이터 출력 방법을 알게 되었지만,
`std::cout`, `<<` 연산자, `std::endl`의 내부 동작 원리는 추후 학습에서 다룬다.

⌨️ scanf를 대신하는 데이터 입력

- 키보드 입력에도 `#include <iostream>` 헤더 파일 선언이 필요하다.
- 키보드 입력에는 `std::cin`과 `>>` 연산자가 사용된다.
- 변수 선언 위치는 함수 내 어디든 가능하다.

> 2_SimpleAdder.cpp

```cpp
#include <iostream>

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

    int val2;
    std::cout << "두 번째 숫자입력: ";
    std::cin >> val2;
```

```
    int result = val1 + val2;
    std::cout << "덧셈결과: " << result << std::endl;

    return 0;
}
```

🔍 관찰결과 1 : 데이터 입력(`std::cin`)

- 기본 형식:

```
std::cin >> 변수;
```

  - 변수 위치에는 입력받은 데이터를 저장할 변수 이름이 온다.
- 예:

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

- 실수 입력:
  - 변수 자료형을 `double`로 선언하면 별도의 포맷 지정 없이 실수 입력 가능
  - `int`형 변수에는 정수 입력, `double`형 변수에는 실수 입력이 자동 진행
- 문자열 입력:
  - C:

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

  - C++:

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

🔍 관찰결과 2 : C++의 지역변수 선언

- C++에서는 함수 내 어디서든 지역 변수 선언 가능
- 대부분의 C++ 컴파일러는 지역 변수 선언 위치에 제한이 없다.
- `for` 문의 초기화 구문에서 변수 선언 가능:

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

- 연속 입력 형식:

```
std::cin >> 변수1 >> 변수2;
```

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

> 3_BetweenAdder.cpp

```cpp
#include <iostream>

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

    std::cout << "두 개의 숫자입력: ";
    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;
}
```

- 앞선 예제와 기본 구조는 동일하며, 입출력 대상이 **문자열**이라는 점만 다르다.
- C++에서도 배열 기반의 문자열 입출력이 가능하다.