1. Hello world로 본 C++

```cpp
#include <iostream>

int main()
{
    std::cout << "Hello world! \n";

    return 0;
}
```

- std는 '네임스페이스(Namespace)' 라고 하며 개념 상 '소속'으로 생각하면 됨.
- :: 은 '범위 지정 연산자' 혹은 '스코프 설정 연산자(Scope resolution operator)' 라고 함.
- cout 은 콘솔 출력을 담당하는 객체
- << 은 '연산자 함수'

▶ std에 속한 cout 객체에 "Hello world! \n" 문자열을 '넘겨(<<)' 문자열을 화면에 출력하라!

2. 인스턴스(instance)

```cpp
int a;
string strdata;
```

- a는 int 자료형에 대한 인스턴스
- strdata는 string 자료형에 대한 인스턴스
- C++에서는 '변수'라는 표현보다 '인스턴스' 라는 표현에 익숙해져야 함.
- 객체지향 프로그래밍환경에서 모든 것을 객체로 표현하고 객체의 형식을 갖는 변수를 인스턴스라고 함.

3. std::cout, std::cin

- 실습 : cin 객체로 입력받고 cout 객체로 출력해보기

```
나이를 입력하세요 : 20
직업을 입력하세요 : 학생
이름을 입력하세요 : 홍길동
당신의 이름은 홍길동이고, 나이는 20살이며, 직업은 학생입니다.
```

```cpp
#include <iostream>
#include <string>
int main()
{
    // 위의 예시처럼 입력받고 출력하는 예제 프로그램을 작성해서 실행시켜보세요~!
    // 문자열 입력은 std::string 객체를 이용
    // std::string strName;
    // std::cin >> strName; 으로 입력 가능
    return 0;
}
```

4. 자료형

- 자료형 : 일정 크기의 메모리에 저장된 정보를 해석하는 방법
- 기본 자료형은 C와 다르지 않음
- C++11 표준에서 추가된 것 확인

자료형	설명	
long long	64비트 정수	
char16_t	16비트 문자 (ex. char16_t ch = u'A';)	유니코드 처리를 위한 자료형
char32_t	32비트 문자 (ex. char32_t ch = u'A';)	
auto	컴파일러가 자동으로 형식을 규정하는 자료형(ex. auto a = 10;)	
decltype(expr)	expr과 동일한 자료형(ex. int x=10; decltype(x)y = 20;)	

■ 변수 선언 및 정의

```
int num = 10; ( C 스타일 선언 및 정의)
int num2(10); ( C++ 스타일 선언 및 정의)
int num3(num2); // 이것도 가능
```

■ auto 예약어

- C++에서 초기값의 형식에 맞춰 선언하는 인스턴스의 형식이 '자동'으로 결정!

```
auto nData = 10;        // nData 형식은 int
auto strName = "Tom";   // strName 형식은 const char*
auto ch = 'A';          // ch 형식은 char
```

5. 메모리 동적할당

- new : 메모리 동적 할당 연산자
- delete : 메모리 해제 연산자

```
int* pData = new int;
delete pData;
```

```
char* mystr = new char[10];
delete[] mystr;
```

```
#include <iostream>
int main()
{
    // 인스터스만 동적으로 생성하는 경우
    int* pData = new int;

    // 초기값을 기술하는 경우
    int* pNewData = new int(10);

    *pData = 5;
    std::cout << *pData << std::endl;
    std::cout << *pNewData << std::endl;

    delete pData;
    delete pNewData;

    return 0;
}
```

```
#include <iostream>
int main()
{
    // 객체를 배열 형태로 동적 생성한다.
    int* arr = new int[5];

    for (int i = 0; i < 5; i++)
        arr[i] = (i + 1) * 10;

    for (int i = 0; i < 5; i++)
        std::cout << arr[i] << std::endl;

    // 배열 형태로 생성한 대상은
    // 반드시 배열 형태를 통해 삭제한다!
    delete[] arr;

    return 0;
}
```