# Chapter 1 : C++에 왔으면 c++의 법을 따릅시다

## 항목 1ㅣ c++의 집합체 파악

C++은 여러 언어의 연합체로 볼 수 있다.

* C : 많은 규칙이 C와 유사하다.
* 객체 지향 개념의 C++ : 클래스를 쓰는 C에 해당되는 부분이다. 생성자 소멸자 캡슐화 상속 다형성 가상함수…
* 템플릿 C++ : 일반화 프로그래밍 부분이다.
* STL : 템플릿의 라이브러리이다. 컨테이너 반복자 알고리즘 함수 객체 등

## 항목 2 ㅣ #define을 쓰려거든 const, enum, inline을 떠올리자

즉 가급적 선행 처리자보다 컴파일러를 더 가까이 하자는 의미이다. 선행 처리자로 선언시 해당 기호식 이름은 컴파일러에겐 보이지 않고 선언된 상수로 보인다. 따라서 해당 기호식 이름이 기호 테이블에 포함되어 있지 않기에 해당 부분에서 에러가 나면 찾기 어려울 수 있다.

Ex) #define ASPECT\_RATIO 1.654 -> const double AspectRatio = 1.654

위와 같이 const 를 사용시 기호테이블에 포함이 되고 최종 코드의 크기도 작아진다. 이는 1.654 라는 사본이 #define의 경우 선행 처리자에 의해 등장한 회수만큼 목적 코드에 들어가지만 const의 경우 몇 번을 쓰더라도 딱 한 개의 사본이 생기기때문이다. 이러한 상수 교체에는 2가지 경우를 조심해야한다.

* 상수 포인터를 정의
  + 상수 정의는 대게 헤더 파일에 넣는 것이 상례이므로 꼭 const로 선언을 해야하고 포인터가 가리키는 대상까지 const로 선언하는 것이 보통이다.ㅜ자세한 사항은 항목 3을 참고.
* 클래스 멤버를 상수로 정의
  + 어떤 상수의 유효 범위를 클래스로 한정하고자 할때는 그 상수를 멤버로 만들어야 하는데, 그 상수의 사본 개수가 한 개를 넘기지 않도록 하고 싶다면 static 멤버로 만들어야한다.

*Ex )*

*class GamePlayer {*

*private :*

*static const int NumTurns = 5;*

*int scores[NumTurns];*

*…*

*};*

위에서 NumTurns는 선언(declaration)되었다. 정의가 아니다. c++ 에서는 정의가 마련되어 있어야 하는게 보통이지만 정적 멤버로 만들어지는 정수류 타입의 클래스 내부 상수는 예외이다. 이들에 대해 주소를 취하지 않는한 정의 없이 선언만 해도 문제가 없다. 이때 클래스 상수의 정의는 구현 파일에 둔다. 헤더 파일에는 두지 않느다. 정의에는 상수의 초기값이 있으면 안되는데 이는 클래스 상수의 초기값은 해당 상수가 선언된 시점에서 바로 주어지기 때문이다.

\*정의와 선언 그리고 초기화

- 선언(declaration)은 코드에 사용되는 어떤 대상의 이름과 타입을 컴파일러에게 알리는 것이다. (내 생각엔 헤더에 해당하는 부분을 의미하는 듯 함)

*Ex)*

*class Widget;*

*extern int x;*

- 정의(definition)은 선언에서 빠진 구체적인 세부사항을 컴파일러에게 제공하는 것이다. 함수나 함수 템플릿에 대한 정의는 그에 대한 코드 본문을 제공하는 것이다.

*Ex)*

*int x;*

*class Widget {*

*public :*

*…*

*};*

- 초기화(initialization)은 어떤 객체에 최초의 값을 부여하는 과정이다. 예를 들어 객체의 경우 생성자에 의해 이루어진다.