C++ 언어 관련 전체 목차 : http://blog.naver.com/tipsware/221028559903

1. C를 품은 C++

보통 언어가 새롭게 만들어지면 그 언어를 상징하는 새로운 이름이 붙여지게 됩니다. 그런데 C++ 언어는 새로운 이름 대신에 하나 더 증가한다는 의미인 Increment 연산자를 C에 붙여서 C++라고 이름 지었습니다. 아무리 봐도 새롭다는 것보다는 개선되었다는 느낌을 주는 이름입니다.

C++ 언어에 '객체지향'을 지원하는 새로운 문법들이 추가되었지만 기본 문법은 C 언어와 동일하기 때문에 C 언어를 그대로 사용할 수 있다는 것을 강조하기 위해서 C++라고 이름 지은 것입니다. 즉, C++ 문법을 모르더라도 이 컴파일러를 C 언어용으로도 사용할 수 있다는 것을 강조해서 C 언어 프로그래머를 좀 더 자연스럽게 C++ 언어로 끌어들이는 효과가 있었다고 생각합니다.

그리고 일단 C++ 언어로 환경을 옮겨오면 조금씩 C++ 문법을 소개해서 자연스럽게 C++ 언어를 사용하게 만들겠다는 전략이 숨어있었습니다. 그리고 C 언어 프로그래머들이 많이 사용하는 ++ 연산자를 사용하여 'New', 'Advance'와 같은 용어보다 더 간결하지만 발전했다는 의미를 더 강하게 전달할 수 있었다고 생각합니다.

결국 C++라는 이름은 C 언어의 확장성과 하드웨어와 친숙하다는 느낌을 그대로 유지하면서 언어의 문법이나 표현이 한 단계 더 발전되었다는 의미를 프로그래머에게 전달하기 위한 이름이라고 생각합니다.

2. C 언어에서 무엇이 ++ 되었는가?

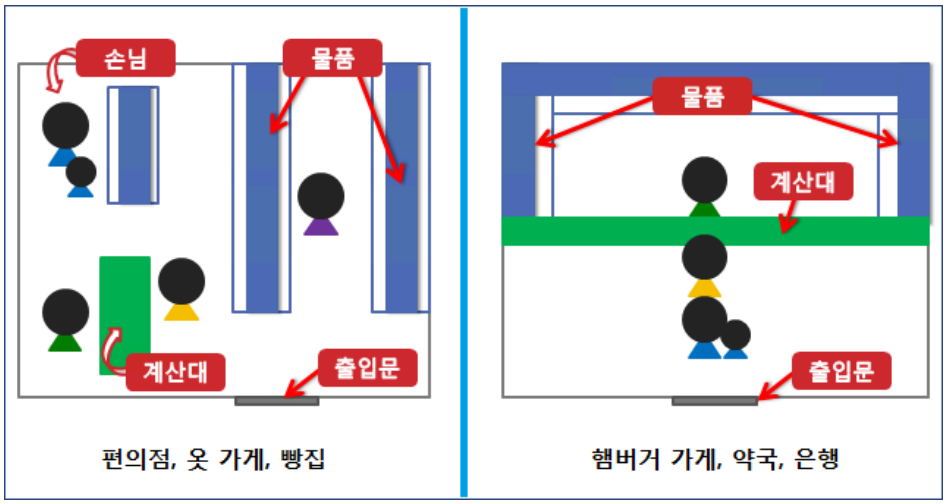
C++ 언어는 이름에 ++가 들어가기 때문에 C 언어를 배운 사람들이라면 C++ 언어가 C 언어에서 무엇이 ++ 되었는지 궁금해합니다. 하지만 C++에서 추가로 제공되는 기능은 그 이름만으로 의미나 역할을 C 언어 프로그래머에게 전달하기 어렵고 생각보다 추가된 문법이나 기술이 너무 많습니다.

그래서 학생들이 **'C++ 언어는 C 언어에서 무엇이 ++ 되었나요?'**라는 질문을 하면 **'객체 지향'이라는 기술이 추가되었습니다.'**라고 간단히 답변하거나 **'지금부터 하나씩 배우게 될** 것입니다.'라고 답변을 할 수밖에 없었습니다.

여전히 저는 처음 C++ 언어를 배우는 사람에게 이해하기 어려운 용어를 나열하는 것은 의미가 없다고 생각합니다. 그래서 무엇이 ++ 되었는지 요약은 생략하고 ++ 된 것들을 하나씩 바로 설명을 하도록 하겠습니다.

3. 프로그램 구성에 대한 철학이 바뀌다

물건을 판매하는 가게는 아래와 같이 크게 두 가지 형태로 나누어집니다. 두 형태 모두 많이 사용되기 때문에 어떤 것이 무조건 더 좋다는 개념으로 말하기는 힘들고, 그냥 판매하는 제품의 특성에 따라 주인이 적합한 가게 형식을 결정한 것이라고 보면 됩니다.



편의점, 옷 가게, 빵집	햄버거 가게, 약국, 은행
자신이 사고 싶은 제품을 선택해서 계산대에서 제품 값을 지불하고 나가는 형태	직원에게 자신이 원하는 제품을 요청하고 제품 값을 지불하면 직원이 제품을 가져와서 전달해주는 형태
손님이 직접 제품에 접근이 가능함	손님이 제품에 접근하는 것이 불가능
여러명의 손님이 동시에 제품에 접근하여 자신이 원하는 제품을 선택할 수 있기 때문에 오른쪽('햄버거 가게')보다 제품 구매가 더 빠르다.	손님은 제품이 어디에 있는지 알 필요가 없기 때문에, 가게 내에서 제품의 이동이 발생하거나 제품에 변화가 생겨도 손님들에게 영향을 주지 않는다. 즉, 가게에서 발생하는 변화가 손님에게 미치는 영향이 별로 없다.
손님들이 물품을 찾기 위해 많이 돌아다녀야 하고 자신이 원하는 것을 찾는데 시간이 걸리거나 어려움을 겪는다. 단골손님의 경우에는 물품을 빨리 찾을 수 있겠지만 가게 내부적인 변화가 생겨서 물품들의 배치가 변경되면 단골손님들도 물건을 찾기 위해 고생해야 한다. 즉, 가게 내부의 배치나 변화가 손님에게 영향을 준다.	손님이 제품을 직접 확인하지 못하고 순차적으로 확인해야 하기 때문에 대기자가 생기게 되고 왼쪽('편의점')보다 제품 구매가 더 느려지게 된다.
물품을 구매할 때 손님과 직원의 역할과 책임이 어디까지 인지 분명하게 구분 짓기 힘들다.	물품을 구매할 때 손님과 직원의 역할과 책임이 분명하게 구분된다.
손님 : 함수 (실행 코드) 물품 : 데이터, 변수, 메모리	
C 언어가 주로 사용하는 방식	**C++ 언어가 주로 사용하는 방식**
데이터를 관리하는 함수(직원)와 데이터를 사용하는 함수(손님)를 구분 짓지 않고 두 가지 성격의 함수가 데이터에 직접 접근하기 때문에 데이터의 형식이 달라지거나 기능이 추가되는 등의 예외가 발생하면 영향을 받는 함수들이 많아진다. 따라서 프로그램이 사용하는 데이터에 변화가 생기면 영향을 받는 함수들의 코드를 일일이 수정해야하기 때문에 개발의 진행률이 높을수록 작업 시간이 많이 소요된다.	데이터와 데이터를 관리하는 함수를 하나로 묶어서 생각하고 이 데이터를 사용하기 위해서는 데이터를 관리하는 함수를 거치도록 만들어서 데이터에 변화가 생기더라도 데이터를 사용하는 함수에는 영향을 미치지 않도록 한다. 따라서 개발 초기에는 데이터를 관리하는 함수를 많이 만들어야 하기 때문에 번거롭지만 프로그램이 사용하는 데이터에 변화가 생겨도 데이터를 관리하는 함수만 영향을 받기 때문에 프로그래머의 스트레스가 줄어든다.

C 언어를 배운 사람들이 C++ 언어를 배울 때 가장 어려워하는 것 중에 하나가 코드 스타일을 바꾸는 것인데 위 표에서 설명한 것과 같이 C 언어에서 주로 사용하는 생각의 방식을 C++ 언어를 사용하는 관점으로 바꾸는 것이 C++ 언어를 이해하는 열쇠가 됩니다. 만약, 이것을 이해하지 못하거나 적응하지 못하면 컴파일러만 C++를 사용할 뿐 결과적으로는 C 언어로 프로그램 하는 것과 큰 차이가 없습니다.

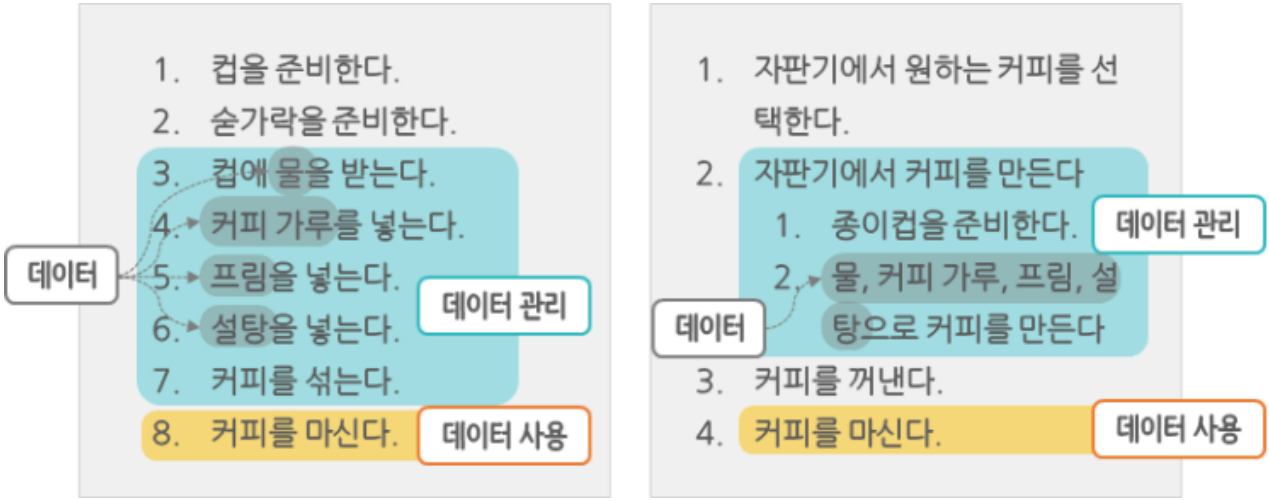
4. 객체란 무엇인가?

객체(Object)라는 개념은 **일상적인 행위나 작업을 표준화 시켜서 하나의 제품으로 만들어 놓은 것**입니다. 예를 들어, 커피 타는 작업을 가지고 '일반적 행위'와 '객체화된 행위'로 구분해서 설명해 보겠습니다.

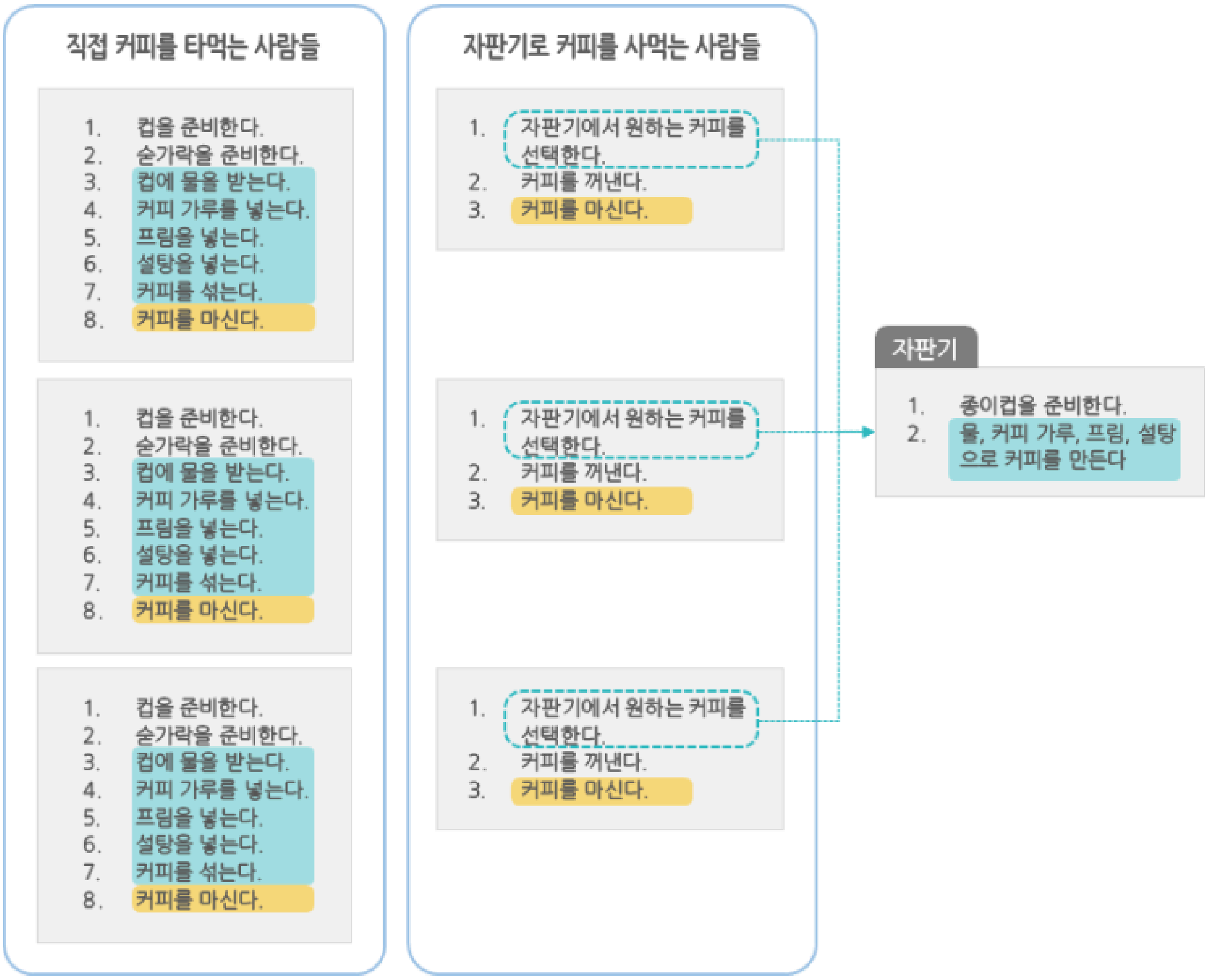
일반적 행위	객체화된 행위
사람이 직접 '커피', '설탕', '프림'을 조합해서 커피를 타서 마시는 행위	커피 자판기를 사용해서 커피를 마시는 행위
세 가지 재료를 조합하는 비율에 따라 맛이 달라지기 때문에 균일한 맛을 유지하기 어렵다.	커피 자판기는 사용자가 선택한 맛에 따라 정확한 비율을 사용해서 커피를 만들기 때문에 맛이 균일하다.
새로운 재료가 추가되면 이 사람은 그 재료를 활용하는 방법을 배워야 한다. 그뿐만 아니라 또 다른 사람이 커피를 마시려면 그 사람도 이 재료를 활용하는 방법을 배워야 합니다. 즉, 새로운 재료는 이 커피를 이용하는 모든 구성원에게 영향을 미친다.	새로운 재료가 추가되면 자판기에 새로운 버튼이 추가될 뿐 자판기를 이용할 사람이 배워야 하거나 알아야 할 사항이 추가되지 않는다.
새로운 것을 배우고 연습을 해야 하기 때문에 초보자는 어려울 수 있다	커피 타는 모든 행위가 자판기 내부에 숨겨져 있기 때문에 초보자도 쉽게 이용 가능하다.
'커피', '설탕', '프림' : 데이터 이 세 가지를 조합하는 행위 : 데이터를 관리하는 작업 커피를 마시는 행위 : 데이터 사용하는 작업	
객체 사용 안 함 - C 프로그래밍	**객체 사용 - C++ 프로그래밍**

모든 기준을 자신이 정해야 함 경험이 풍부하면 잘 하겠지만 그렇지 않다면 데이터와 작업을 잘못 구분하여 상황 변화에 대처하기 어려운 경우가 많이 생김	**객체가 기준이 됨** 객체를 만드는 작업과 객체를 사용하는 작업을 구분하기 때문에 상황 변화에 대처하기 쉬움
데이터 관리(커피 타기)와 데이터를 사용하는 작업(커피를 마시기)이 구분되지 않고 사용하는 작업에 관리 작업이 조금씩 포함되어 코드의 중복이 발생한다. (모든 사람이 커피를 탄다)	데이터를 관리하는 행위와 사용하는 행위가 정확하게 구분되어 서로에게 미치는 영향이 적다. (커피는 자판기만 탄다)
데이터에 변화가 생기면 변경해야 할 곳이 많아져서 작업량이 늘어나고 작업을 하다가 실수할 확률도 높아진다. (모든 사람이 커피를 타는 행위를 하기 때문에 새로운 재료가 추가되었을 때 모든 사람이 그 재료의 사용법을 알아야 한다)	데이터에 변화가 생기면 데이터를 관리하는 작업만 수정하면 된다. (새로운 재료가 추가되는 경우 자판기만 변경하면 작업이 완료된다)

[각 행위에 대한 상세 정의]



[전체 행위에 대한 중복]



이처럼 **커피 타는 행위를 하나의 제품으로 만든 자판기가 '객체'에 해당**합니다. 따라서 객체 개념을 사용하면 자신이 원하는 행위를 역할에 따라서 정확히 구분하고 행위자와 소비자의 연결고리(인터페이스)를 단순하게 만들어서 활용성과 재사용성을 높일 수 있습니다.

5. '객체 지향(OOP)'이란 무엇인가?

'객체 지향(OOP, Object-oriented programming)'이라는 말의 의미는 '객체 개념을 사용해서 프로그램을 하겠다'라는 뜻입니다. 즉, 프로그램에서 필요한 행위들을 일반화 시켜서 객체로 만들고 이 객체를 사용하는 형태로 프로그래밍을 하겠다는 것입니다.

C 언어로 프로그래밍하는 경우에는 보통 필요한 행위를 바로 함수화합니다. 하지만 C++의 경우에는 객체를 만들기 위해 고민을 하다 보면 자신이 보지 못했던 데이터의 공통점이나 의미 없는 행위를 구분하게 되어 더 효율적으로 작업할 수 있습니다. 또한 프로그래밍 작업이 '객체를 만드는 작업'과 '객체를 사용하는 작업'으로 나누어지기 때문에, 작업할 때 고민하는 범위가 줄어드는 효과도 있습니다. 예를 들어, 자판기를 만드는 사람은 자판기를 만드는 고민만 하면 되고 자판기를 이용하는 사람은 이용하는 방법에 대해서만 고민하면 됩니다.

그리고 이렇게 만들어진 객체는 다른 프로그램에서 동일한 행위(기능)가 필요한 경우 그대로 다시 사용할 수 있기 때문에 프로그램의 개발 속도에도 큰 영향을 미칩니다. 예를 들어, 커피 자판기를 다른 곳에서도 필요로 한다면 기존 커피 자판기를 그대로 다시 만들어서 사용(객체 소스를 그대로 복사해서 옮김)하면 된다는 것입니다.

C 언어도 함수를 정의하여 코드를 다시 사용할 수 있지만 객체의 재사용성에 비하면 많이 부족합니다. 예를 들어, 커피세트와 커피 타는 매뉴얼을 만들어서 보급하면 누군가 따라다니면서 교육하는 부담은 줄어들겠지만 자판기를 설치하는 것이 더 효과적이라는 것입니다. 물론 C 언어를 잘 하는 사람이라면 함수를 멋지게 설계하여 믹스 커피 같은 개념을 만들어서 편리하게 만들 수도 있

지만 이것은 경험 많은 프로그래머나 가능한 이야기입니다.

6. C++ 언어는 '객체 지향' 언어이다.

C 언어는 함수를 기반으로 작업하는 절차 지향 언어이고 **C++ 언어는 객체를 만들고 그 객체를 기반으로 프로그래밍하는 '객체 지향' 언어입니다.** 그래서 C++ 언어에는 객체를 만들 수 있는 문법과 이 객체를 관리하고 좀 더 쉽게 재사용될 수 있도록 도와주는 여러 가지 문법들이 제공됩니다.

따라서 앞으로 배울 C++ 문법은 결국 '객체 지향'을 추구하는 여러 가지 개념을 배우는 과정이 될 것입니다. 결국 아무 생각 없이 하던 행위들을 일정한 형식으로 일반화 시켜서 객체를 만드는 연습과 이 객체를 어떻게 하면 잘 활용할 수 있는지에 대한 여러 가지 문법을 배우게 될 것입니다. 그래서 C++ 언어를 배운다는 것은 '객체 지향' 프로그래밍을 배우는 것과 같습니다.