C++ 강좌 04회 : 객체와 클래스

김성엽 카페매니저 + 구독 1:1 채팅
2022.05.11. 03:07 조회 420

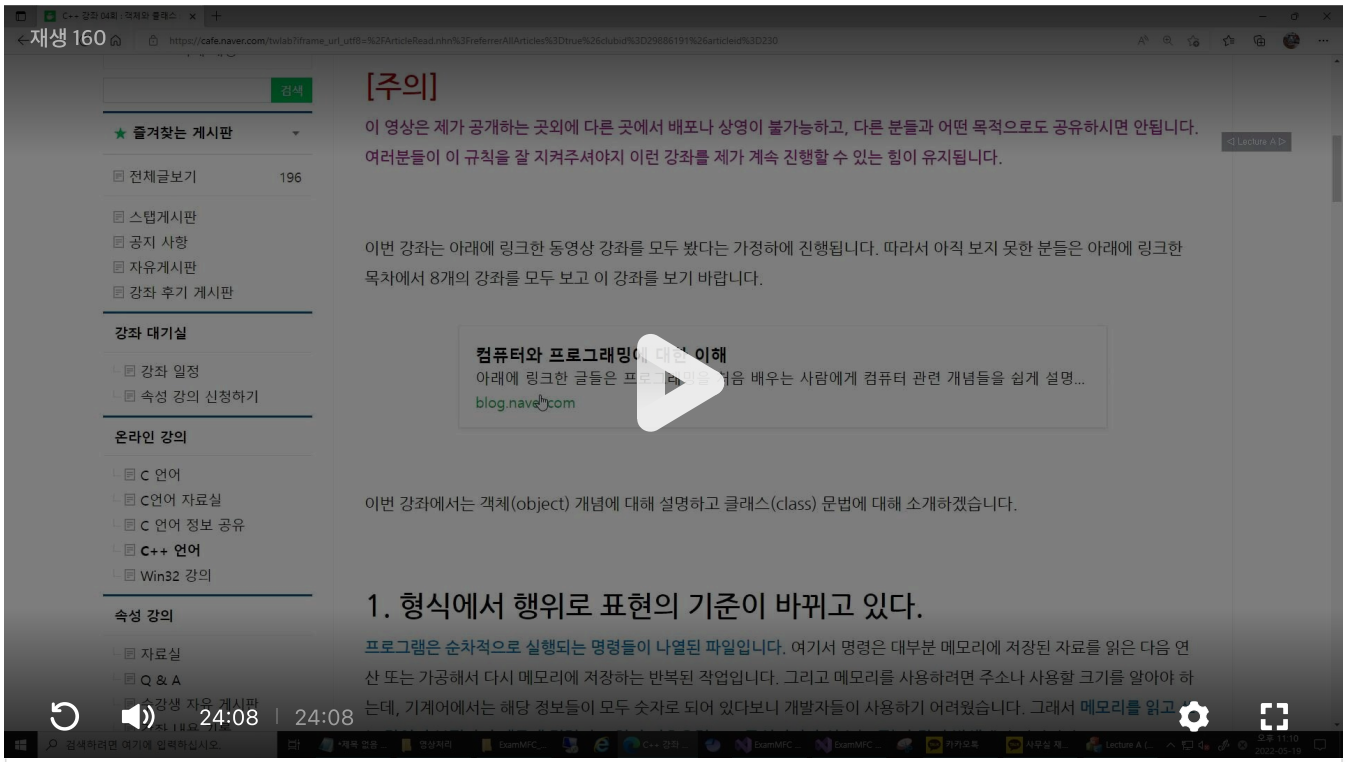
[주의]

이 영상은 제가 공개하는 곳외에 다른 곳에서 배포나 상영이 불가능하고, 다른 분들과 어떤 목적으로도 공유하시면 안됩니다. 여러분들이 이 규칙은 잘 지켜주셔야지 이런 강좌를 제가 계속 진행할 수 있는 힘이 유지됩니다.

이번 강좌는 아래에 링크한 동영상 강좌를 모두 봤다는 가정하에 진행됩니다. 따라서 아직 보지 못한 분들은 아래에 링크한 목차에서 8개의 강좌를 모두 보고 이 강좌를 보기 바랍니다.

컴퓨터와 프로그래밍에 대한 이해
아래에 링크한 글들은 프로그래밍을 처음 배우는 사람에게 컴퓨터 관련 개념들을 쉽게 설명하기 위해 적은 ...
blog.naver.com

이번 강좌에서는 객체(object) 개념에 대해 설명하고 클래스(class) 문법에 대해 소개하겠습니다.



C++ 강좌 04회 : 객체와 클래스

1. 형식에서 행위로 표현의 기준이 바뀌고 있다.

프로그램은 순차적으로 실행되는 명령들이 나열된 파일입니다. 여기서 명령은 대부분 메모리에 저장된 자료를 읽은 다음 연산 또는 가공해서 다시 메모리에 저장하는 반복된 작업입니다. 그리고 메모리를 사용하려면 주소나 사용할 크기를 알아야 하는데, 기계어에서는 해당 정보들이 모두 숫자로 되어 있다보니 개발자들이 사용하기 어려웠습니다. 그래서 메모리를 읽고 쓰는 작업이 분편하기 때문에 명령어 표현이 비효율적으로 구성되거나 실수(오류)가 많이 발생했던 것입니다.

이로인해 프로그래밍 언어를 개발하는 사람들은 개발자가 메모리를 편하게 사용하는 것이 중요하다는 것을 깨닫게 되고 이 편리함이 개발자의 실수도 줄여 준다는 것을 알게 됩니다. 그래서 프로그래밍 언어는 자료형(Data Type)의 시대를 맞이하게 됩니다.

컴퓨터 기술이 발전하고 활용도가 높아짐에 따라 프로그램에서 사용하는 데이터(메모리)의 크기나 구성이 다양해져서 개발자가 필요로하는 자료형을 프로그래밍 언어가 모두 제공하는 것이 불가능해졌습니다. 그래서 사용자가 자신이 원하는 형태로 자료형은 정의해 사용하는 사용자 정의 자료형(ex 구조체)이라는 문법이 추가된 것입니다.

그리고 IT 산업의 발전으로 프로그램의 규모가 커지면서 프로그램을 개발하는 비용보다 유지 보수 비용이 더 많은 시대로 접어들게 됩니다. 즉, 기능 변경이나 데이터 변화에 잘 대처할 수 있는 프로그램을 만들어야하는 시대가 온 것입니다. 그래서 구조화된 프로그래밍 언어(ex C 언어)들이 등장했고 이 언어들은 함수라는 문법은 제공하는데 이 문법이 변화에 잘 대처할 수 있는 프로그램은 만드는 핵심적인 역할을 하게 됩니다.

어떻게 보면 이 때부터 데이터의 형식보다는 데이터를 사용하는 행위가 더 강조되는 시대가 시작되었다고 볼 수 있습니다. 즉, 데이터의 형식을 이해해서 사용하던 시대에서 데이터의 형식은 몰라도 자신이 원하는 작업은 해주는 함수만 호출하면 원하는 결과를 얻은 수 있는 시대로 접어든 것입니다.

그래서 개발자들은 기본적으로 제공되거나 다른 개발자가 정의한 다양한 데이터의 형식에는 점점 더 무관심해지고 자신에게 필요한 함수를 찾는 시대로 접어들게 됩니다. 즉, **데이터의 형식보다는 자신이 하고 싶은 행위가 더 중요한 시대**가 된 것입니다. 그런데 함수를 사용하는 개발자 입장에서 가장 불편한 점은 함수를 사용하려면 해당 함수가 어떤 자료형을 매개 변수로 사용하는지에 대한 이해가 필요하다는 것입니다.

이 문제점을 해결하기 위해 프로그래밍 언어는 구조체와 함수를 통합하는 개념을 사용했고 이 개념이 객체(object)입니다. **객체는 작업에 대한 행위를 표준화시켜 개발자가 객체에 어떤 형식의 데이터가 존재하는지 몰라도 해당 객체가 제공하는 함수를 호출해서 원하는 결과를 얻을 수 있도록 도와주는 기술**입니다. 즉, 객체는 데이터의 형식보다는 데이터를 사용하는 행위를 중요하게 생각하는 기술이고 이런 기술을 지향하는 프로그래밍을 **객체 지향 프로그래밍(OOP, Object-oriented programming)**이라고 합니다.

다른 관점에서 생각해보면 이제 **기능을 제공하는 개발자와 기능을 사용하는 개발자의 시대**가 온 것입니다. 객체 기술을 사용하여 기능을 제공하는 개발자는 데이터 형식을 정의하고 이 데이터를 사용하는 함수를 작성하기 때문에 구조체와 함수를 한 번에 정의하도록 문법 형식이 달라졌을 뿐 이전 시대와 달라진 것이 별로 없습니다. 하지만 기능을 사용하는 개발자의 입장에서는 자신이 원하는 기능을 어떤 객체가 가지고 있고 그 객체에 어떤 함수를 사용하면 되는지만 알면 프로그램을 만들 수 있게 되었습니다.

결국 모든 개발자가 고급 기술자가 될 수 없다는 것을 인정하고 필요한 만큼 이해하고 사용할 수 있는 개발자도 필요하다는 것을 인정하게 된 것입니다. 그래서 점점 더 형식보다는 행위를 중요시하는 프로그래밍 언어가 많이 생기는 것입니다.

2. class 문법의 추가

기능을 구현할 때 데이터의 형식은 최대한 숨기고 행위를 강조하는 객체 개념이 초반에는 자료형 강화의 성격으로 struct 문법에 적용되었습니다. 하지만 C++ 언어는 C 언어의 문법을 거의 동일하게 사용하는데 struct 문법 확장이 기존 C 언어의 struct 문법과 키워드는 동일하면서 기능이 다르기 때문에 개발자들 사이에 혼선이 생겼을 것입니다. 그래서 C++ 언어는 불필요한 혼선과 표준 논쟁을 피하기 위해 struct 문법과 거의 유사하지만 **객체를 정의하는 새로운 문법으로 class를 추가**했습니다.

class 문법이 추가되었다고 해서 C++ 언어의 struct 문법이 축소되어 C 언어 문법과 동일하게 다시 변경된 것은 아닙니다. 지금도 **C++ 언어의 struct 문법은 기능적으로는 class와 거의 유사하게 유지**되고 있습니다. 그래서 일부 개발자들은 class 문법을 사용하지 않고 struct 문법을 class처럼 사용하는 경우도 있고 두 문법을 섞어서 사용하기도 합니다.

개발자의 개별적인 스타일은 존중하지만 저는 객체를 정의하는 class 문법이 있는데 struct 문법을 동일한 개념으로 사용하는 것은 개발자의 표현을 제한한다고 생각합니다. 예를 들어, 단순한 자료 표현을 하고 싶을 수도 있는데 struct 문법도 객체 개념을 표현하는 문법이니 객체 개념을 유지할수 있도록 형식을 지켜달라는 불필요한 논쟁이 생길수도 있다는 뜻입니다. 그래서 **저는 교육할 때 C++ 언어에서도 struct 문법은 C 언어의 struct 문법과 동일하게 사용하고 객체 개념은 class 문법으로만 표현하는 것을 권장**합니다.

따라서 지금까지는 struct 문법을 사용해서 설명했는데 **지금부터는 struct 대신 class 문법을 사용해서 설명**할 것이며 기존에 사용했던 강좌나 소스에서 struct 대신 class를 사용해도 개념과 결과는 동일합니다. 다만 차이가 있다면 struct 문법을 사

용할 때는 접근 지정자를 생략했을 때 public 접근 지정자가 생략된 것이지만, class 문법은 사용하게 되면 접근 지정자를 생략하면 private 접근 지정자가 생략된 것입니다.

3. 객체와 class의 관계

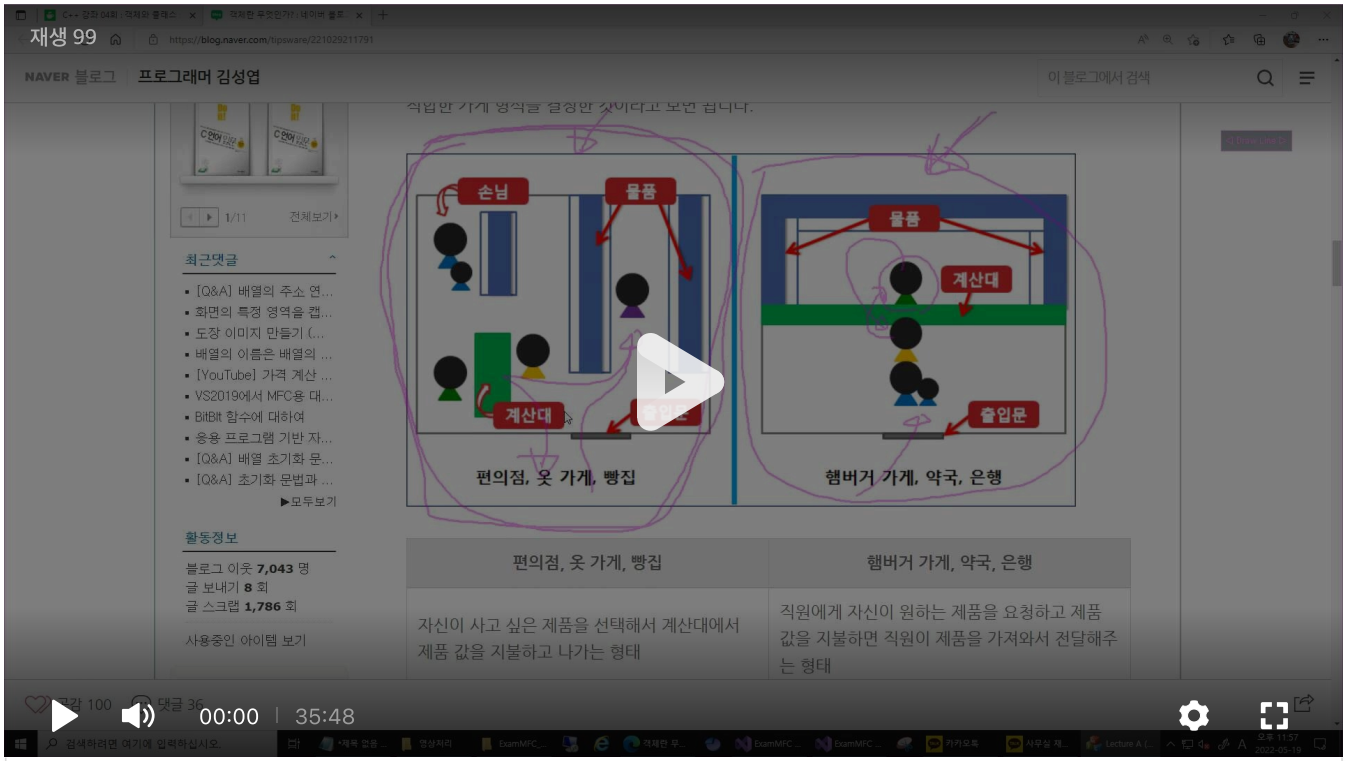
class 문법은 객체를 정의하기 위한 설계도 같은 개념입니다. 예를 들어, 냉장고 설계도를 가지고 냉장고를 만들었다면 냉장고 설계도가 class 이고 만들어진 냉장고가 객체가 됩니다. 하지만 class를 가지고 객체를 만든다고 해서 class 자체가 객체라고 이야기 할 수는 없습니다. class는 사용할 물건을 만들기 위한 준비물일뿐 물건 자체가 될 수는 없습니다.

따라서 C++ 언어를 사용하는 개발자들은 'class를 사용해서 객체를 생성(선언)한다.'라는 표현을 사용합니다. 그리고 이것은 아래와 같이 자료형으로 변수를 선언하는 것과 동일한 형태를 가집니다.

```
class MyData
{
private:
    int m_data;    // 멤버 변수 선언

public:
    // 외부에서 m_data 변수에 값을 저장할 때 사용하는 함수
    void SetData(int a_data)
    {
        if(a_data < 0) a_data = -a_data;  // 음수면 양수로 변경
        m_data = a_data;
    }

    // 외부에서 m_data 변수에 저장된 값을 얻을 때 사용하는 함수
    int GetData()
    {
        return m_data;
    }
};

int main()
{
    int a, b, c;  // 4바이트 정수형 변수 a, b, c를 선언한다.
    MyData data, temp, result;  // MyData class로 data, temp, result 객체를 선언한다.

    return 0;
}
```

그리고 객체에 대한 좀더 일반적인 설명은 아래에 링크한 자료를 참고하기 바랍니다. 객체 개념은 C++ 언어를 이해하는 기본적인 개념이기 때문에 아래에 링크한 글은 자주 보는 것이 좋습니다.

객체란 무엇인가?
C++ 언어 관련 전체 목차 : http://blog.naver.com/tipsware/221028559903 1. C를 품은 C++ 보통 언어…
blog.naver.com



4_1회차. 객체란 무엇인가?

클린봇이 악성 댓글을 감지합니다. 설정

댓글 등록순 최신순 관심글 댓글 알림

카일 3
다시 한번 개념 복습하고 갑니다~ 강의 잘 들었습니다. ^^
2022.11.21. 12:09 답글쓰기

김성엽 작성자

수고하셨습니다
2022.11.21. 13:43 답글쓰기

dh221009
댓글을 남겨보세요
등록