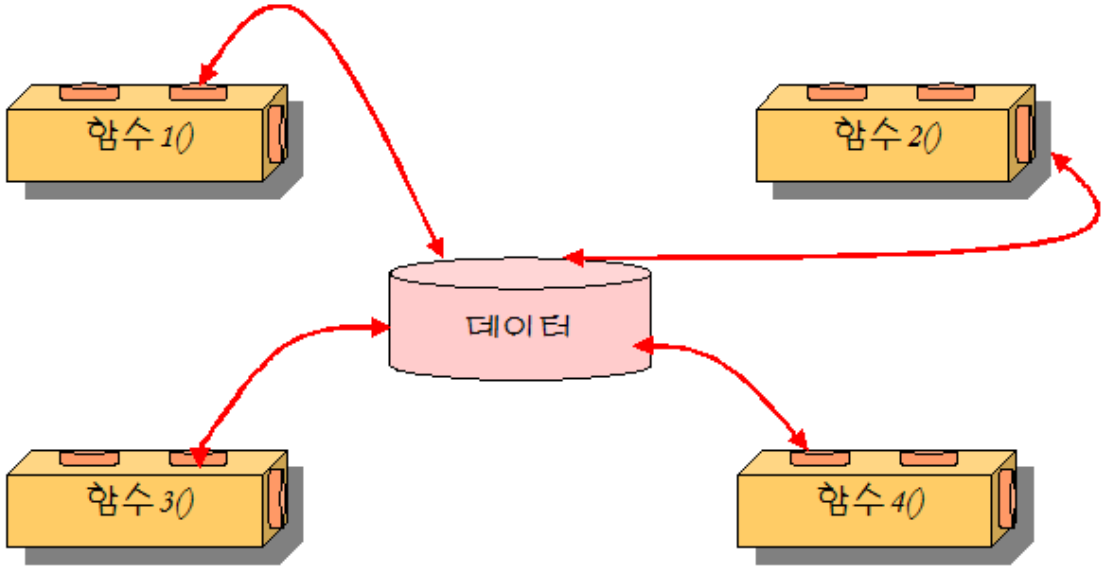
c with classes : C++

국민대학교 임은진

• 절차 지향 프로그래밍(Procedural Programming)

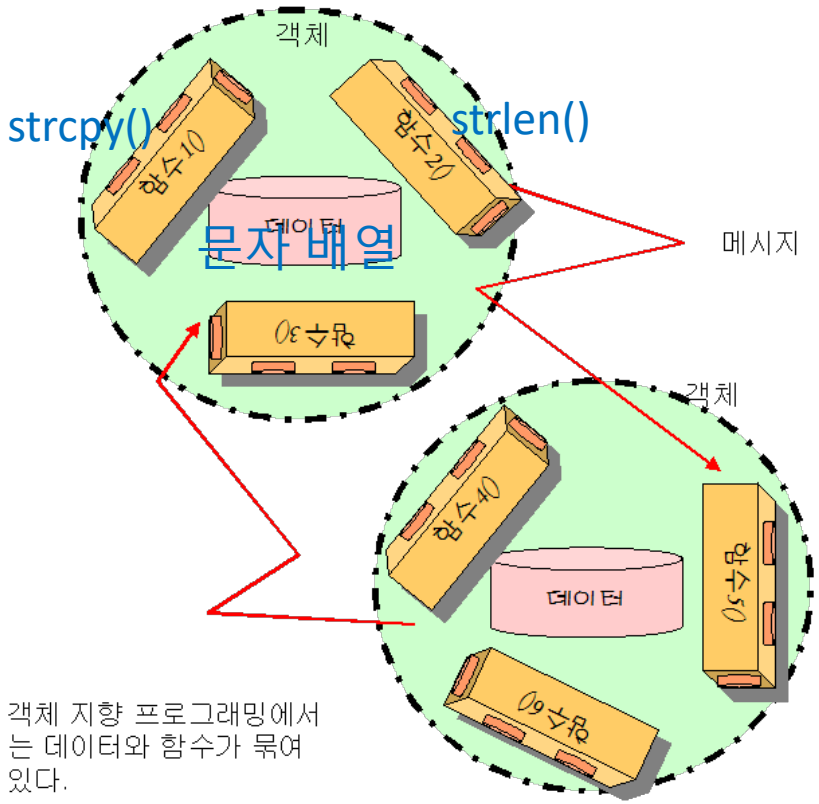
- 문제를 해결하는 절차를 중요하게 생각하는 소프트웨어 개발 방법.
- 이들 절차는 모두 함수라는 단위로 묶이게 된다.



절차 지향 프로그래밍에서는 데이터와 함수가 묶여 있지 않다.

- 객체 지향 프로그래밍(Object-Oriented Programming)

  - 데이터와 함수를 하나의 덩어리로 묶어서 생각하는 방법이다.
  - 데이터와 함수를 객체로 묶는 것을 캡슐화(encapsulation)라고 부른다.

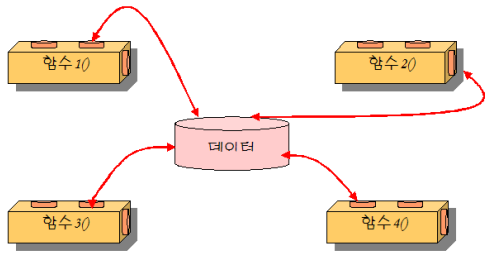


객체 지향 프로그래밍에서는 데이터와 함수가 묶여 있다.

자동차 경주 게임의 예

절차 지향

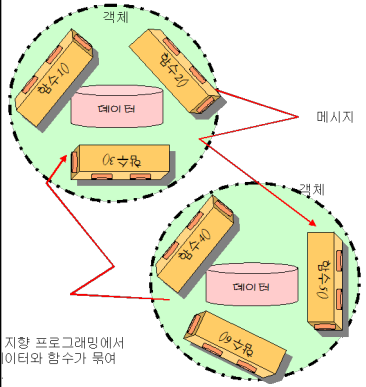
```
struct Car {
    int speed;
    int gear;
    char *pcolor;
};

void init(Car& c, char *color);
void start(Car& c);
void stop(Car& c);
int get_speed(Car& c);
void set_speed(Car& c, int speed);

int main()
{
    Car car;
    init(car, "red");
    start(car);
    set_speed(car, 60);
    stop(car);
    return 0;
}
```



절차 지향 프로그래밍에서는 데이터와 함수가 묶여 있지 않다.

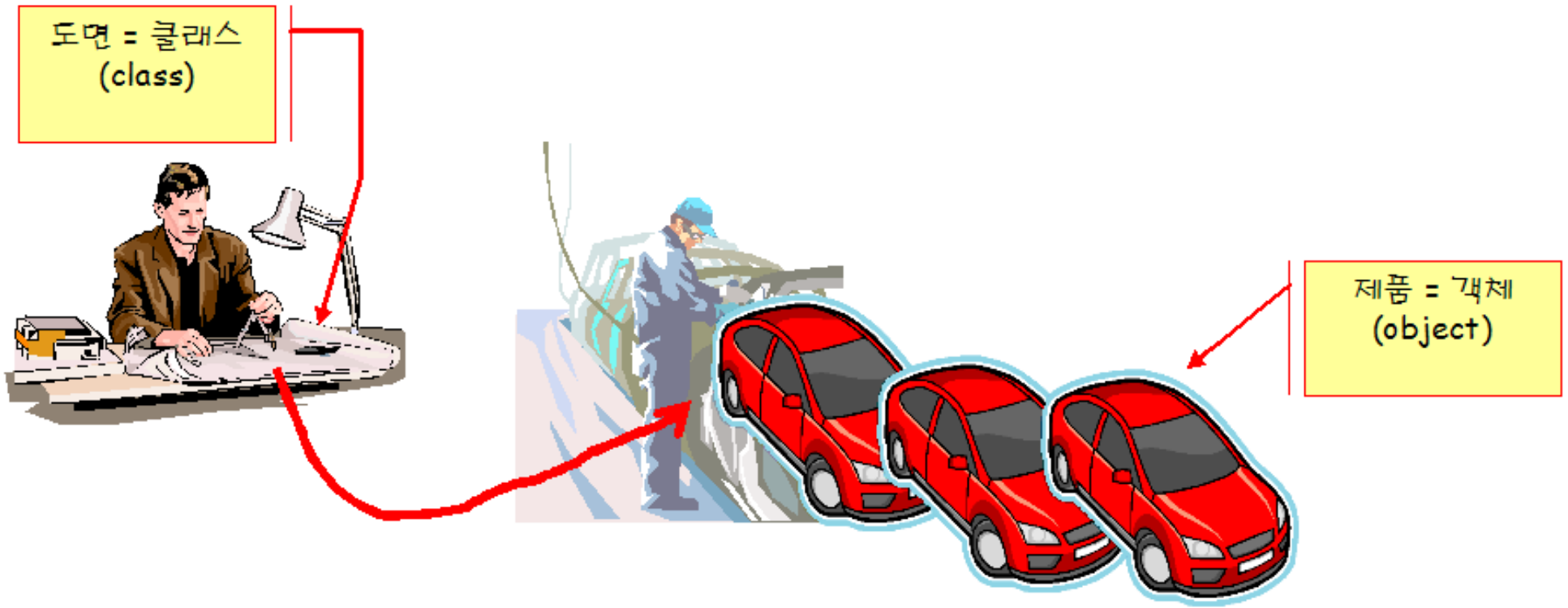
객체 지향

```
class Car {
    int speed;
    int gear;
    char *pcolor;

public:
    void init(char *color);
    void start();
    void stop();
    int get_speed();
    void set_speed(int speed);
};

int main()
{
    Car car;
    car.init("red");
    car.start();
    car.set_speed(60);
    car.stop(car);
    return 0;
}
```



클래스

- 클래스(class): 객체를 만드는 설계도
- 클래스로부터 만들어지는 각각의 객체를 특별히 그 클래스의 인스턴스**(instance)**라고도 한다.



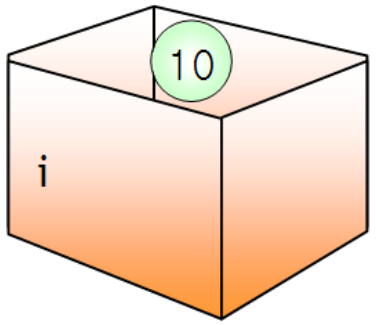
객체

type 선언이유: 메모리를 얼마나 쓰고 어떻게 해석해야 하는지 알려줌

- class : user-defined type
- instance(object) of the class : 그 타입으로 선언된 변수

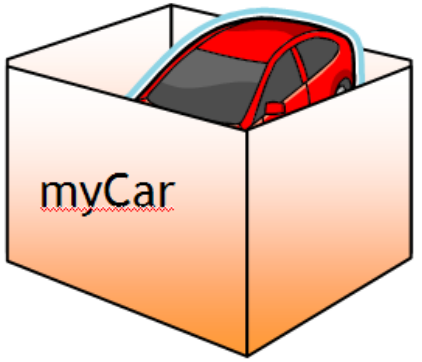
- int 타입의 변수를 선언하는 경우

  int i;



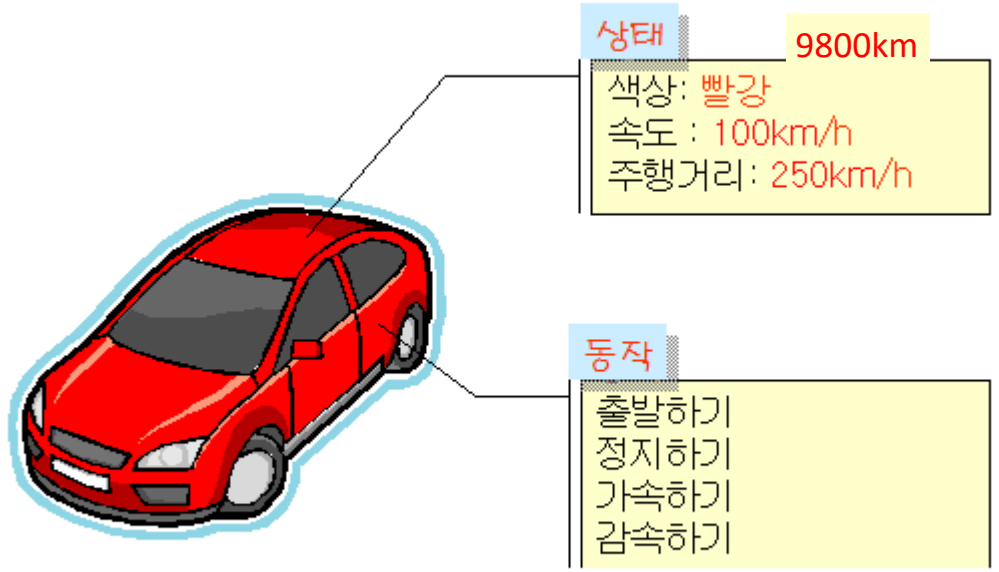
- Car 타입의 변수를 선언하면 객체가 생성된다.

  Car myCar;

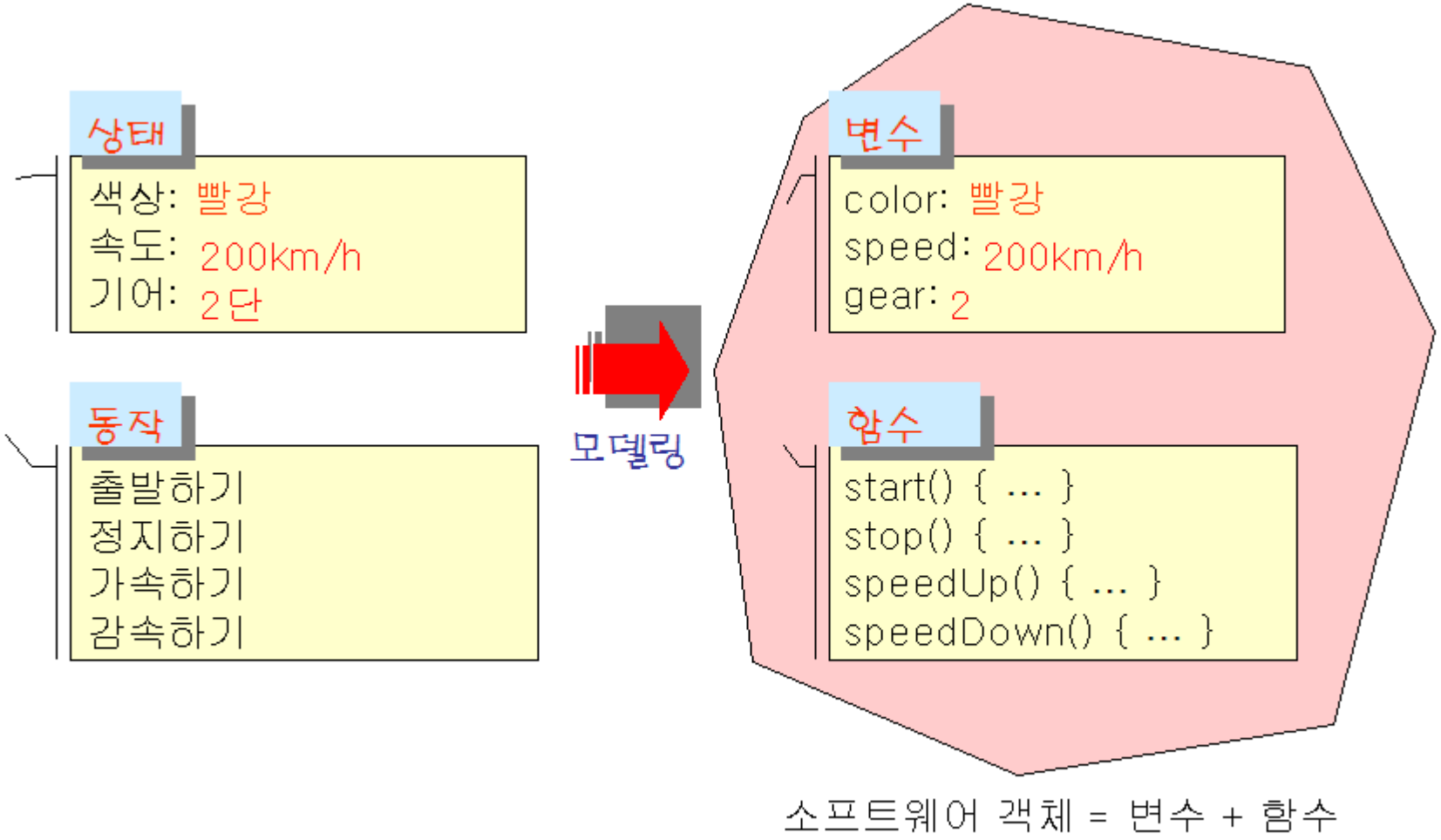


객체란?

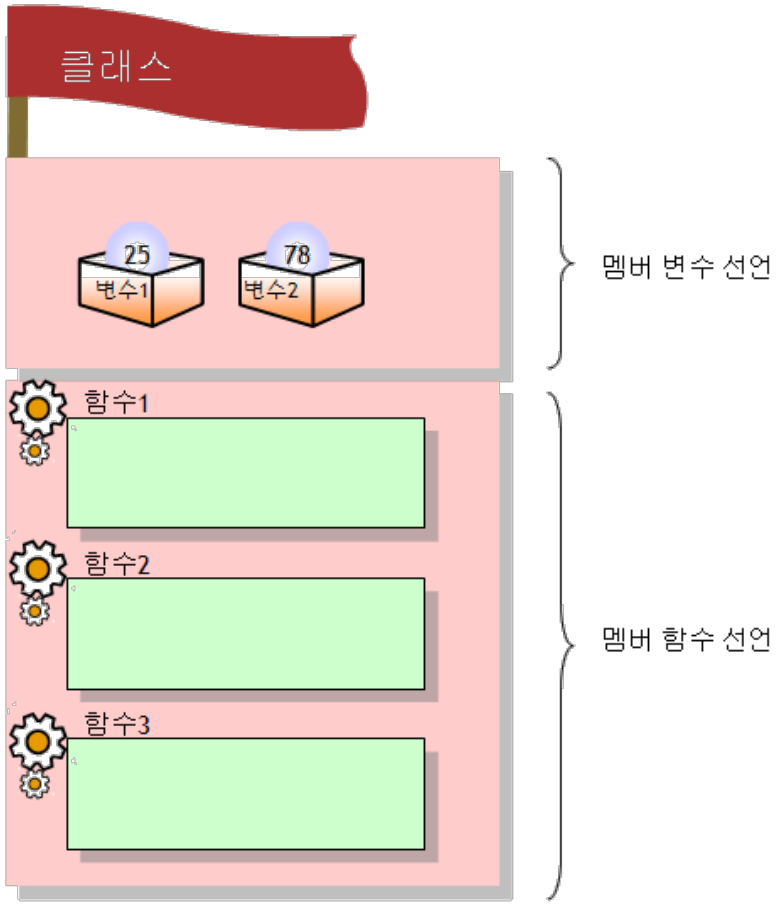
- 객체(object)는 상태와 동작을 가지고 있다.
- 객체의 상태(state)는 객체의 특징값(속성)이다.
- 객체의 동작(behavior) 또는 행동은 객체가 취할 수 있는 동작



멤버 변수와 멤버 함수


상태
색상: 빨강
속도: 200km/h
기어: 2단
동작
출발하기
정지하기
가속하기
감속하기
모델링
변수
color: 빨강
speed: 200km/h
gear: 2
함수
start() { … }
stop() { … }
speedUp() { … }
speedDown() { … }
소프트웨어 객체 = 변수 + 함수

클래스의 구성



- 클래스(class)는 객체의 설계도라 할 수 있다.

- 클래스는 멤버 변수와 멤버 함수로 이루어진다.
- 멤버 변수는 객체의 속성을 나타낸다.
- 멤버 함수는 객체의 동작을 나타낸다.

클래스 정의의 예

```
class Car {

public:
    // 멤버 변수 선언
    int speed; // 속도
    int gear; // 기어
    string color; // 색상

    // 멤버 함수 선언
    void speedUp() { // 속도 증가 멤버 함수
            speed += 10;
    }

    void speedDown() { // 속도 감소 멤버 함수
            speed -= 10;
    }
};
```

멤버 변수 정의!

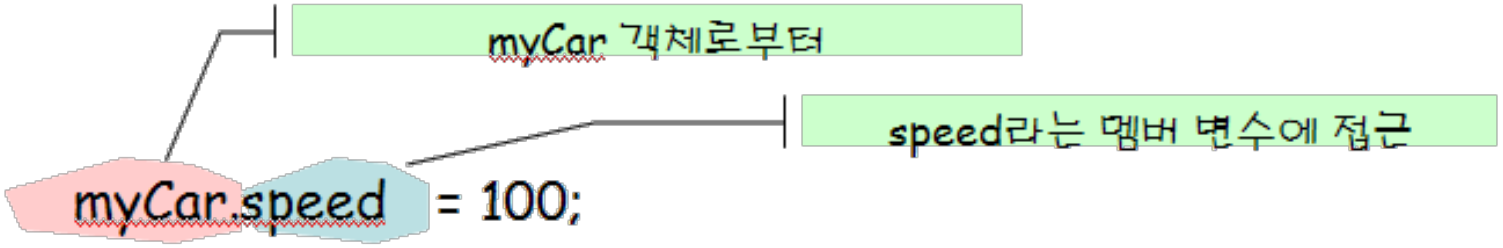
멤버 함수 정의!

↳ local, global 은 Car 이 local, global 중 어떤걸로 선언되었는지에 따라 결정

객체의 사용 operator .

- 객체를 이용하여 멤버에 접근할 수 있다.

```
Car myCar;
```



이들 멤버 변수에 접근하기 위해서는 도트(.) 연산자를 사용한다.

```
myCar.speed = 100;
myCar.speedUp();
myCar.speedDown();
```

객체도 pointer, reference 사용이 가능

```
Car myCar;
Car *pCar = &myCar; // Car *pCar; pCar = &myCar;
```

-> 연산자 사용이 가능

```
myCar.speed = 100;
pCar->speed = 120; // (*pCar).speed = 120;
pCar->speedUp();
```

예제

```
#include <iostream>
#include <string>
using namespace std;

class Car {
public:
	// 멤버 변수 선언
	int speed; // 속도
	int gear; // 기어
	string color; // 색상

	// 멤버 함수 선언
	void speedUp() { // 속도 증가 멤버 함수
		speed += 10;
	}

	void speedDown() { // 속도 감소 멤버 함수
		speed -= 10;
	}
};
```

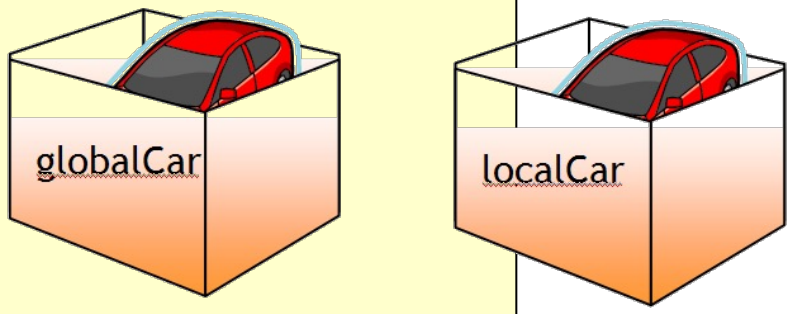
객체(object)

- 속성 또는 상태(state)
- 동작 또는 행위(behavior):
  출발, 정지, 가속, 감속, 방향 전환

구 분		2.0 VVT
전 장 (mm)		4,505
전 폭 (mm)		1,775
전 고 (mm)		1,480
윤 거	전 (mm)	1,545(1,530)
	후 (mm)	1,540(1,525)
축간거리 (mm)		2,650
엔진형식		2.0 VVT
배 기 량 (cc)		1,975
최고출력 (ps/rpm)		143/6,000 (M/T) 134/6,000 (A/T)
최대토오크 (kg·m/rpm)		19.0/4,600 (M/T) 18.4/4,600 (A/T)
연료탱크용량 (ℓ)		53

using an object

```
Car globalCar;                                  // ① 전역 객체
int main()
{
      Car localCar;                             // ② 지역 객체

      globalCar.speed = 100;;
      localCar.speed = 60;
      localCar.color = "white";

      cout << "현재 global 차의 속도는 " << globalCar.speed << endl;
      cout << "현재 local 차의 속도는 " << localCar.speed << endl;

      return 0;
}
```




```
현재 global 차의 속도는 100
현재 local 차의 속도는 60
계속하려면 아무 키나 누르십시오 . . .
```

구조체 struct

- 여러 개의 field 들을 가지는 user-defined type in C
- class 와 유사 in C++

```
struct   _point {
            int x;
            int y;
};
struct   _point p1;              // C 언어 방식
_point p2;                       // C++ 언어 방식
```

class vs struct

내부 default가 private임

내부 default가 public 임