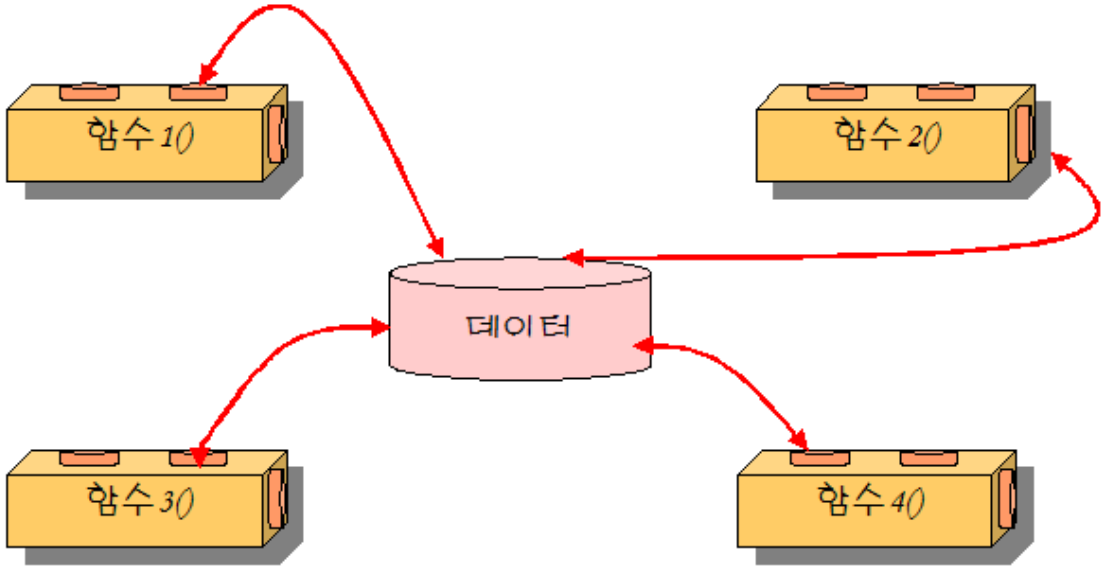
c with classes : C++
a string class example

2023

국민대학교 소프트웨어학부

• 절차 지향 프로그래밍(Procedural Programming)

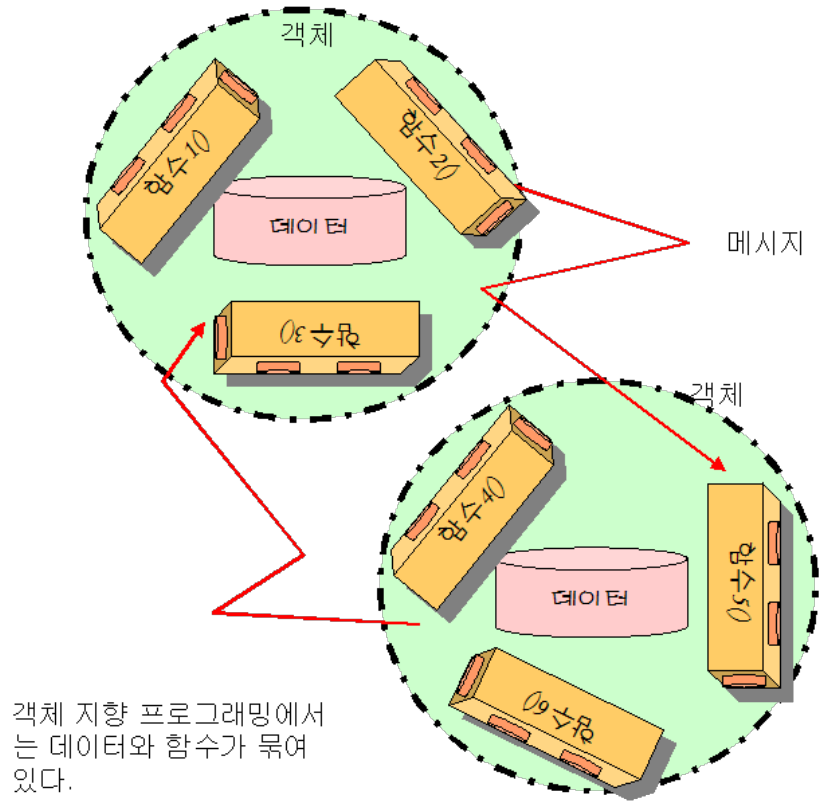
- 문제를 해결하는 절차를 중요하게 생각하는 소프트웨어 개발 방법.
- 이들 절차는 모두 함수라는 단위로 묶이게 된다.



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

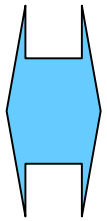
```
struct Car {
    int speed;
    int gear;
    char *pcolor;
};

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;
}
```

객체 지향



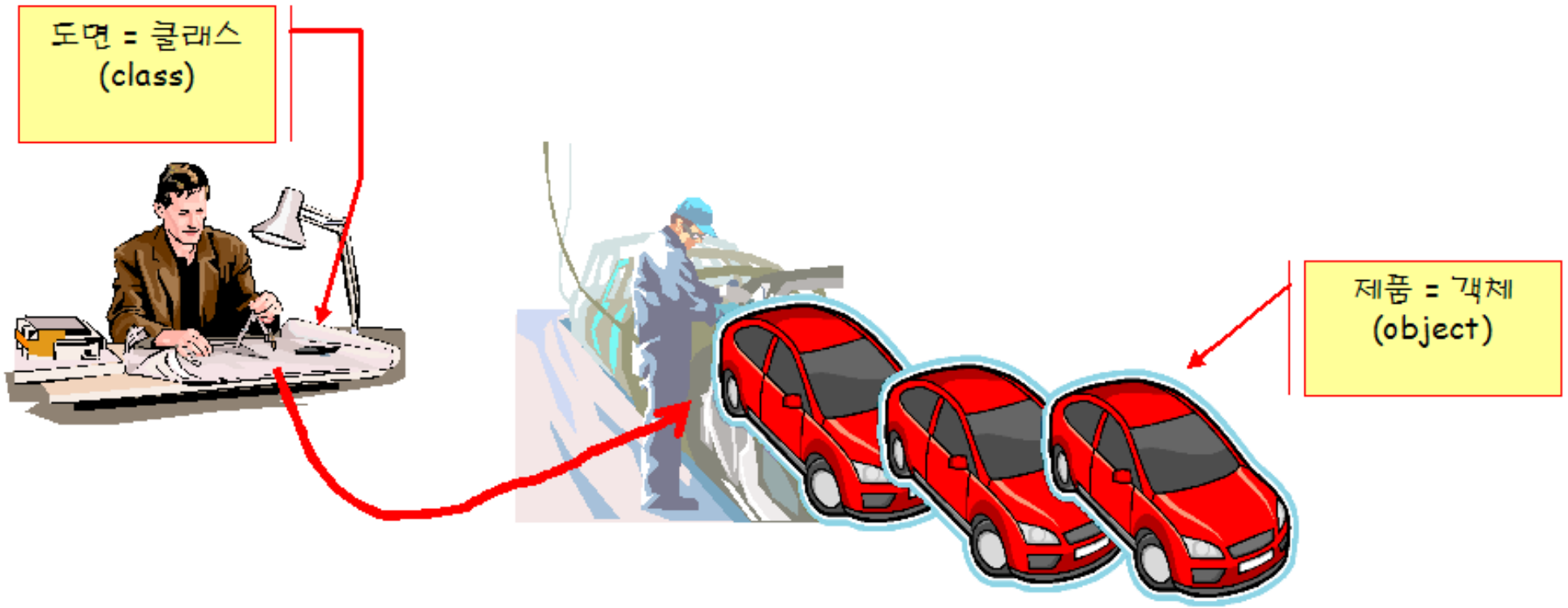
```
class Car {
    int speed;
    int gear;
    char *pcolor;

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

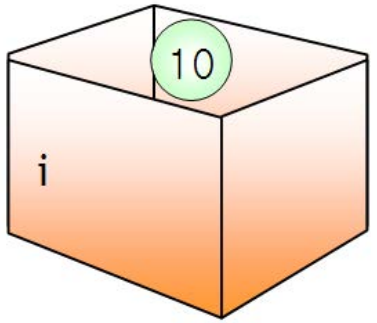
- 클래스(class): 객체를 만드는 설계도
- 클래스로부터 만들어지는 각각의 객체를 특별히 그 클래스의 인스턴스**(instance)**라고도 한다.



객체

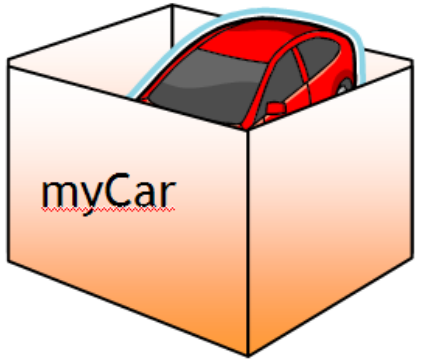
- int 타입의 변수를 선언하는 경우

  int i;



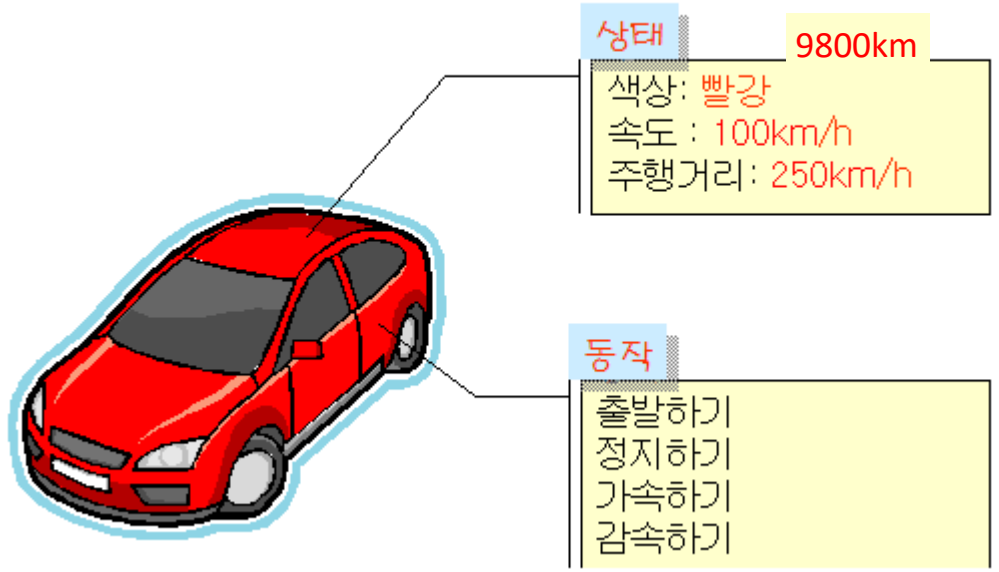
- 클래스도 타입으로 생각하면 된다.
- Car 타입의 변수를 선언하면 객체가 생성된다.

  Car myCar;

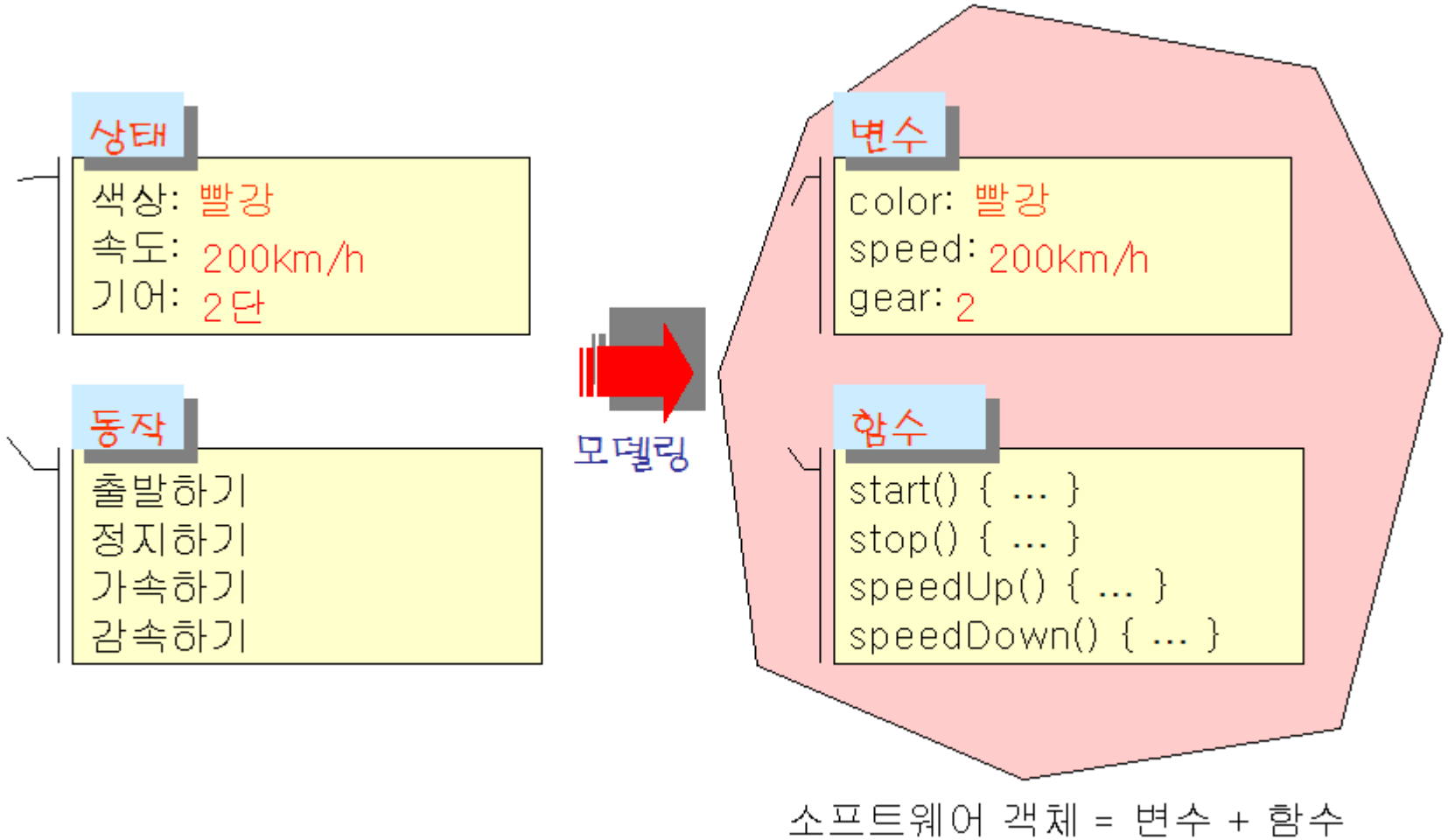


객체란?

- 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

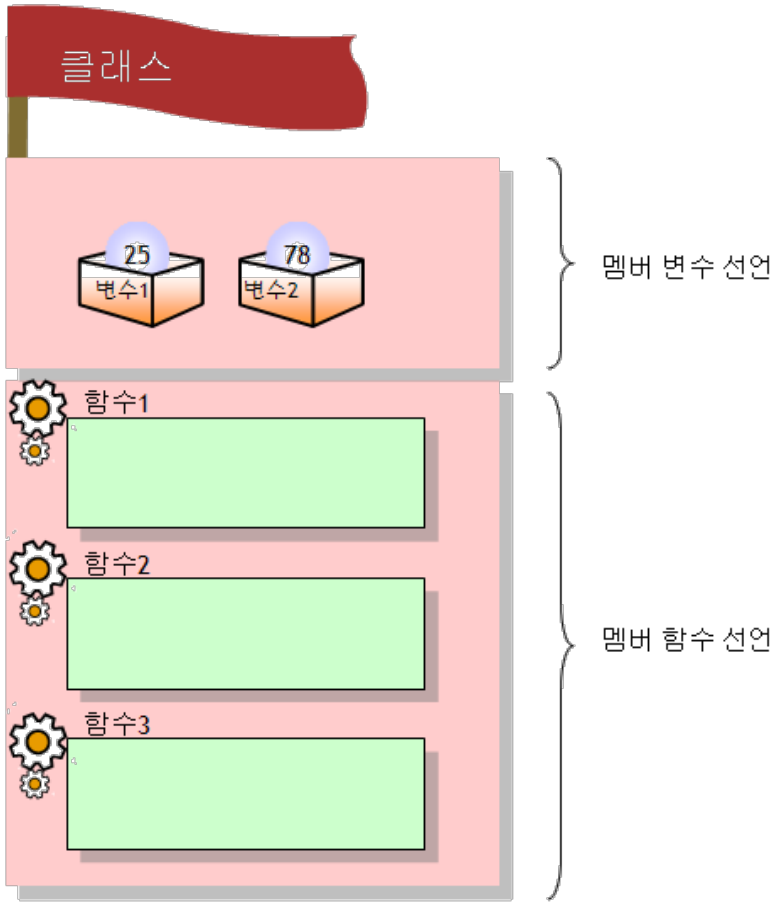


멤버 변수와 멤버 함수



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;
    }
};
```

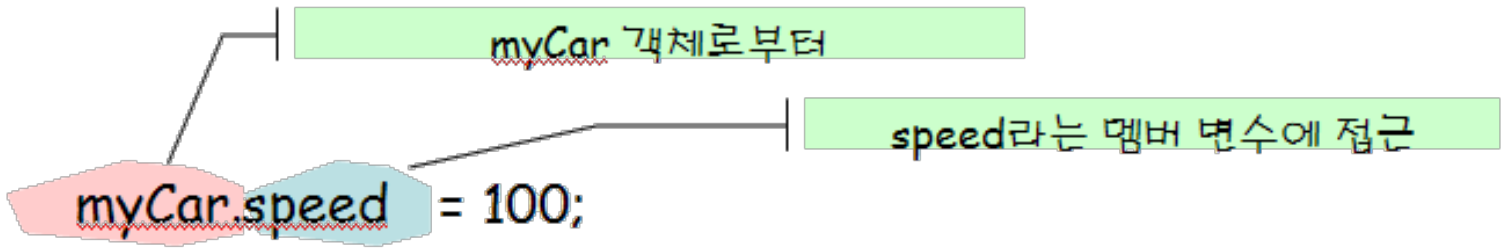
멤버 변수 정의!

멤버 함수 정의!

객체의 사용

- 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

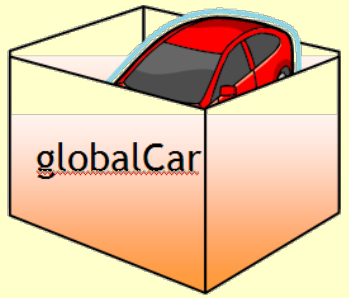
using an object

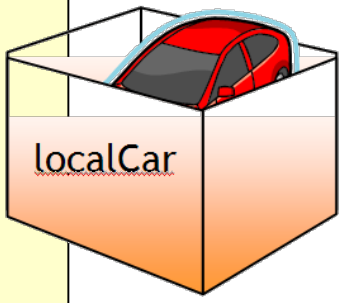
```
Car globalCar;                                  // ① 전역 객체
int main()
{
      Car localCar;                             // ② 지역 객체

      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

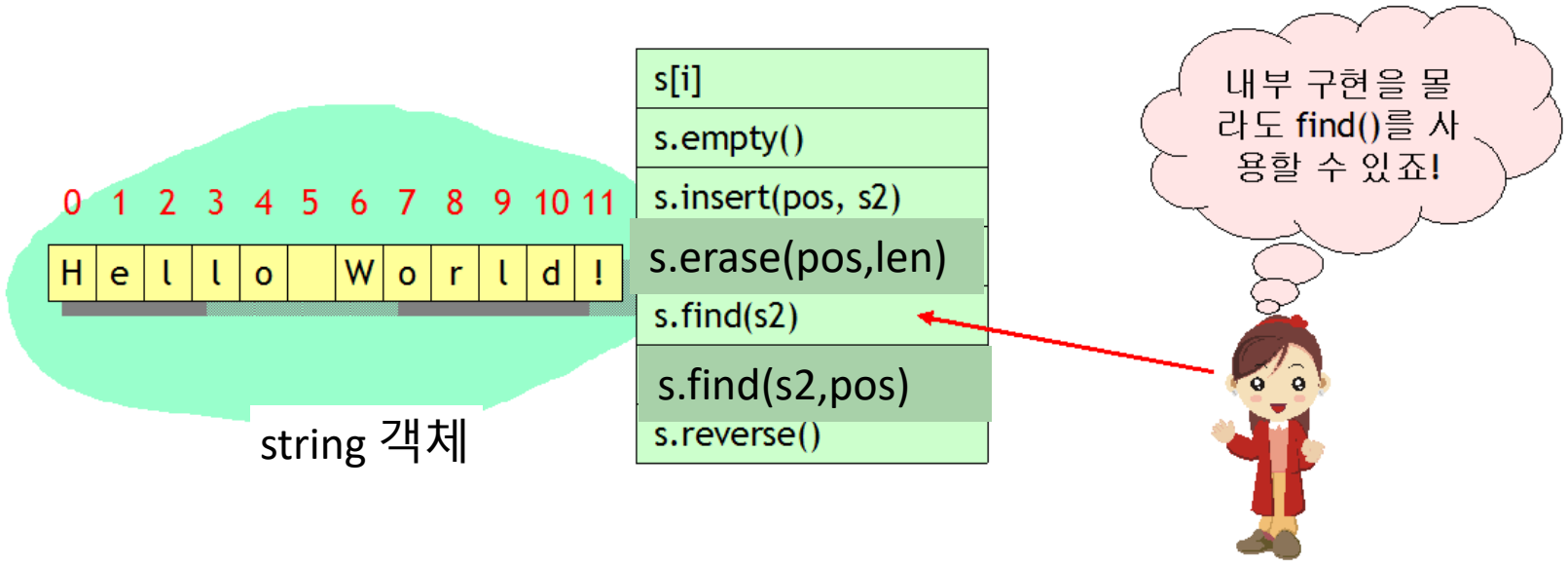
- 여러 개의 field 들을 가지는 user-defined type in C
- class 와 유사 in C++

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

클래스 사용의 예 : string 클래스

- C++에서는 문자열을 나타내는 클래스 string을 제공한다.

#include <string>



class

std::string


```
typedef basic_string<char> string;
```

String class

Strings are objects that represent sequences of characters.

The standard `string` class provides support for such objects with an interface similar to that of a standard container of bytes, but adding features specifically designed to operate with strings of single-byte characters.

The `string` class is an instantiation of the basic_string class template that uses `char` (i.e., bytes) as its *character type*, with its default char_traits and allocator types (see basic_string for more info on the template).

Note that this class handles bytes independently of the encoding used: If used to handle sequences of multi-byte or variable-length characters (such as UTF-8), all members of this class (such as length or size), as well as its iterators, will still operate in terms of bytes (not actual encoded characters).

Member types

member type	definition
value_type	char
traits_type	char_traits<char>
allocator_type	allocator<char>
reference	char&
const_reference	const char&
pointer	char*
const_pointer	const char*
iterator	a random access iterator to char (convertible to const_iterator)
const_iterator	a random access iterator to const char
reverse_iterator	reverse_iterator<iterator>
const_reverse_iterator	reverse_iterator<const_iterator>
difference_type	ptrdiff_t
size_type	size_t

fx Member functions

(constructor)	Construct string object (public member function)
(destructor)	String destructor (public member function)
operator=	String assignment (public member function)

Iterators:

begin	Return iterator to beginning (public member function)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
cbegin C++11	Return const_iterator to beginning (public member function)
cend C++11	Return const_iterator to end (public member function)
crbegin C++11	Return const_reverse_iterator to reverse beginning (public member function)
crend C++11	Return const_reverse_iterator to reverse end (public member function)

Capacity:

size	Return length of string (public member function)
length	Return length of string (public member function)
max_size	Return maximum size of string (public member function)
resize	Resize string (public member function)
capacity	Return size of allocated storage (public member function)
reserve	Request a change in capacity (public member function)
clear	Clear string (public member function)
empty	Test if string is empty (public member function)
shrink_to_fit C++11	Shrink to fit (public member function)

Element access:

operator[]	Get character of string (public member function)
at	Get character in string (public member function)
back C++11	Access last character (public member function)
front C++11	Access first character (public member function)

Modifiers:

operator+=	Append to string (public member function)
append	Append to string (public member function)
push_back	Append character to string (public member function)
assign	Assign content to string (public member function)
insert	Insert into string (public member function)
erase	Erase characters from string (public member function)
replace	Replace portion of string (public member function)
swap	Swap string values (public member function)
pop_back C++11	Delete last character (public member function)

String operations:

c_str	Get C string equivalent (public member function)
data	Get string data (public member function)
get_allocator	Get allocator (public member function)
copy	Copy sequence of characters from string (public member function)
find	Find content in string (public member function)
rfind	Find last occurrence of content in string (public member function)
find_first_of	Find character in string (public member function)
find_last_of	Find character in string from the end (public member function)
find_first_not_of	Find absence of character in string (public member function)
find_last_not_of	Find non-matching character in string from the end (public member function)
substr	Generate substring (public member function)
compare	Compare strings (public member function)

```
static const size_type npos = -1;
```

This is a special value equal to the maximum value representable by the type `size_type`. The exact meaning depends on context, but it is generally used either as end of string indicator by the functions that expect a string index or as the error indicator by the functions that return a string index.

fx Member constants

npos	Maximum value for size_t (public static member constant)

fx Non-member function overloads

operator+	Concatenate strings (function)
relational operators	Relational operators for string (function)
swap	Exchanges the values of two strings (function)
operator>>	Extract string from stream (function)
operator<<	Insert string into stream (function)
getline	Get line from stream into string (function)

public member function

std::string::find

<string>

C++98 C++11

```
string (1)      size_t find (const string& str, size_t pos = 0) const;
c-string (2)    size_t find (const char* s, size_t pos = 0) const;
buffer (3)      size_t find (const char* s, size_t pos, size_t n) const;
character (4)   size_t find (char c, size_t pos = 0) const;
```

Find content in string

Searches the string for the first occurrence of the sequence specified by its arguments.

When *pos* is specified, the search only includes characters at or after position *pos*, ignoring any possible occurrences that include characters before *pos*.

Notice that unlike member find_first_of, whenever more than one character is being searched for, it is not enough that just one of these characters match, but the entire sequence must match.

Parameters

`str`
Another string with the subject to search for.

`pos`
Position of the first character in the string to be considered in the search.
If this is greater than the string length, the function never finds matches.
Note: The first character is denoted by a value of `0` (not `1`): A value of `0` means that the entire string is searched.

`s`
Pointer to an array of characters.
If argument *n* is specified *(3)*, the sequence to match are the first *n* characters in the array.
Otherwise *(2)*, a null-terminated sequence is expected: the length of the sequence to match is determined by the first occurrence of a null character.

`n`
Length of sequence of characters to match.

`c`
Individual character to be searched for.

size_t is an unsigned integral type (the same as member type `string::size_type`).

Return Value

The position of the first character of the first match.
If no matches were found, the function returns string::npos.

size_t is an unsigned integral type (the same as member type `string::size_type`).

string 클래스의 멤버 함수 중 일부

멤버 함수	설명
s[i]	i번째 원소
s.size()	문자열의 길이를 반환 (null 문자(='\0') 제외)
s.clear()	문자열을 지우고 null 문자열(="")로 만듦
s.empty()	s가 비어있으면 true 반환
s.insert(pos, s2)	s의 pos 위치에 문자열 s2를 삽입
s.erase(pos, len)	s의 pos 위치부터 len만큼의 문자들을 삭제
s.find(s2)	s에서 문자열 (혹은 문자)s2가 발견되는 첫번째 인덱스를 반환
s.find(s2, pos)	s의 pos 위치부터 문자열 (혹은 문자) s2가 발견되는 첫번째 인덱스를 반환

찾는 문자열(혹은 문자)가 없으면 string::npos (= -1) 을 반환

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

int main(){

  string s1;
  string s2("I");
  string s3 = s2;
  string s4 = "you";

  cout << "s1 : \"" << s1 << "\" size " << s1.size() << endl;
  cout << "s2 : \"" << s2 << "\" size " << s2.size() << endl;
  cout << "s3 : \"" << s3 << "\" size " << s3.size() << endl;
  cout << "s4 : \"" << s4 << "\" size " << s4.size() << endl;

  s1 = s2 + " and " + s4;
  cout << "s1 : \"" << s1 << "\" size " << s1.size() << endl;
  s1.clear();
  cout << "s1 : \"" << s1 << "\" size " << s1.size() << endl;
  cout << "is s1 empty? " << s1.empty() << endl;

  s4[0] = 'Y';
  cout << "s4 : \"" << s4 << "\" size " << s4.size() << endl;
```

escape sequence

```
s1 : "" size 0
s2 : "I" size 1
s3 : "I" size 1
s4 : "you" size 3
s1 : "I and you" size 9
s1 : "" size 0
is s1 empty? 1
s4 : "You" size 3
```

```
25    s1 = "abcdefghijklmnopqrstuvwxyz";
26    int i = s1.find('c');
27    cout << "c is the " << i << "-th alphabet.\n";
28    i = s1.find('c', i+1);
29    cout << "the next c is the " << i << "-th alphabet.\n";
30    i = s1.find("efg");
31    cout << "efg starts at " << i << "-th alphabet.\n";
32    cout << s2 << " < " << s4 << " : " << (s2 < s4) << endl;
33    cout << s2 << " == " << s4 << " : " << (s2 == s4) << endl;
34
35    cout << "write a word to insert : ";
36    cin >> s3;
37    s1.insert(0, s3);
38    cout << "s1 : \"" << s1 << "\" size " << s1.size() << endl;
39    s1.erase(1, 3);
40    cout << "s1 : \"" << s1 << "\" size " << s1.size() << endl;
41
42    return 0;
43 }
```

```
c is the 2-th alphabet.
the next c is the -1-th alphabet.
efg starts at 4-th alphabet.
I < You : 1
I == You : 0
write a word to insert : cprogram
s1 : "cprogramabcdefghijklmnopqrstuvwxyz" size 34
s1 : "cgramabcdefghijklmnopqrstuvwxyz" size 31
```

예제: wheel of fortune

wheel of fortune

- 빈칸으로 구성된 문자열이 주어지고 사용자는 문자열에 들어갈 글자들을 하나씩 추측해서 맞추는 게임이다.

```cpp
#include <iostream>
#include <string>
#include <cstdlib>
#include <ctime>
using namespace std;

#define DICT_LEN 3
#define MAX_TRIES 6

int main(){

  string dictionary[] = {"space", "wheel", "program"};
  srand(time(NULL));
  string prob = dictionary[rand()%DICT_LEN];
  int length = prob.length();
  int tries = 0;
  string answer(length, '-');

  cout << "current state : " << answer << endl;
```

```
current state : -------
```

```
20  while (tries < MAX_TRIES && answer != prob){
21     char c;
22     cout << "guess a letter : ";
23     cin >> c;
24     if (answer.find(c) != string::npos){
25        cout << c << " is guessed previously.\n";
26        continue;
27     }
28     int pos = prob.find(c);
29     if (pos == string::npos){
30        cout << c << " is not in the target string.\n";
31        tries++;
32        continue;
33     }
34     do {
35        answer[pos] = c;
36        pos = prob.find(c, pos+1);
37     } while (pos != string::npos);
38     cout << "current state : " << answer << endl;
39     if (answer == prob){
40        cout << "You WON!!!\n";
41        break;
42     }
43  }
44  if (tries >= MAX_TRIES){
45     cout << "You lost : the answer was " << prob << endl;
46  }
```

```
current state : -------
guess a letter : e
e is not in the target string.
guess a letter : p
current state : p------
guess a letter : r
current state : pr--r--
guess a letter : m
current state : pr--r-m
guess a letter : o
current state : pro-r-m
guess a letter : g
current state : progr-m
guess a letter : a
current state : program
You WON!!!
```

```
20  while (tries < MAX_TRIES && answer != prob){
21     char c;
22     cout << "guess a letter : ";
23     cin >> c;
24     if (answer.find(c) != string::npos){
25        cout << c << " is guessed previously.\n";
26        continue;
27     }
28     int pos = prob.find(c);
29     if (pos == string::npos){
30        cout << c << " is not in the target string.\n";
31        tries++;
32        continue;
33     }
34     do {
35        answer[pos] = c;
36        pos = prob.find(c, pos+1);
37     } while (pos != string::npos);
38     cout << "current state : " << answer << endl;
39     if (answer == prob){
40        cout << "You WON!!!\n";
41        break;
42     }
43  }
44  if (tries >= MAX_TRIES){
45     cout << "You lost : the answer was " << prob << endl;
46  }
```

```
current state : -----
guess a letter : r
r is not in the target string.
guess a letter : h
h is not in the target string.
guess a letter : p
current state : -p---
guess a letter : p
p is guessed previously.
guess a letter : a
current state : -pa--
guess a letter : e
current state : -pa-e
guess a letter : i
i is not in the target string.
guess a letter : o
o is not in the target string.
guess a letter : u
u is not in the target string.
guess a letter : t
t is not in the target string.
You lost : the answer was space
```