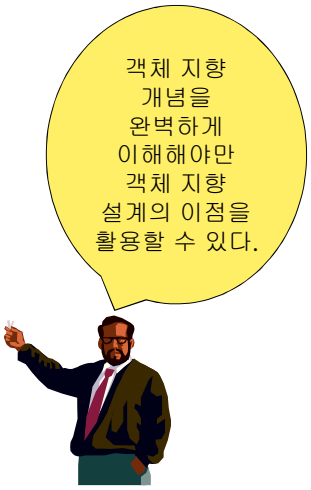
C++ Espresso

제4장 객체 지향 소개

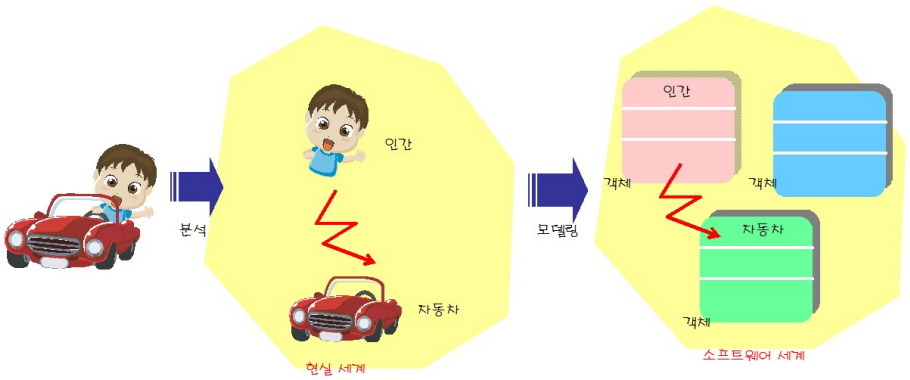
이번 장에서 학습할 내용

- 객체지향이란?
- 객체
- 메시지
- 클래스
- 객체 지향의 장점
- string 클래스



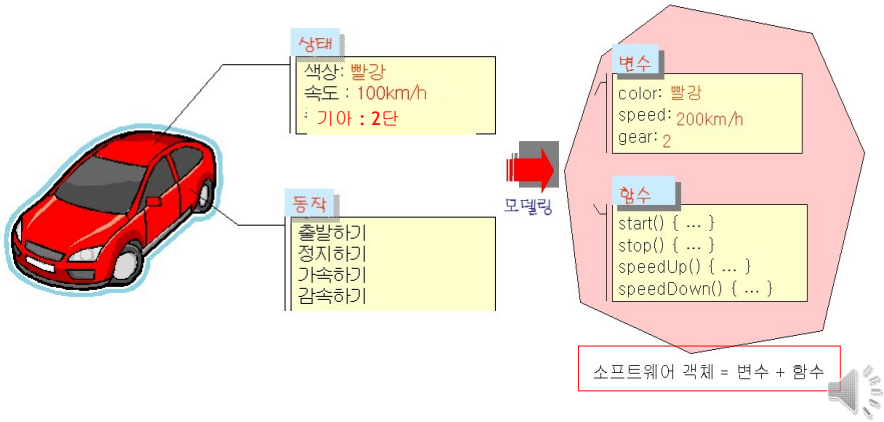
객체 지향이란 ?

- 실제 세계를 모델링하여 소프트웨어를 개발하는 방법



객체란?

- 객체(object)는 상태와 동작을 가지고 있다.
- 객체의 상태(state)는 객체의 특징값(속성)이다. → 멤버변수
- 객체의 동작(behavior) 또는 행동은 객체가 취할 수 있는 동작 → 멤버함수
- 객체는 변수와 함수를 포함한다.



객체란?

- C 방식
  - 변수(값 저장), 함수(특정 임무 수행) 각각 따로
- C++, 객체지향 방식
  - 객체 → 변수(값 저장) + 함수 (특정 임무 수행) 를 묶어 사용

객체지향, 클래스

- 객체지향 프로그래밍, 클래스 객체 ?
  - 기존 프로그래밍 C
    - 기존에 사용했던 type(데이터 형) : int, float, char, bool …
    - 2와 4를 더하여 6을 구했다.
    - (대)명사는 변수(정수형 type 사용) → 2, 4, 6 을 정수형 변수에 저장
    - 정수 관련 프로그램은 int 사용, 실수 관련 프로그램은 float 사용

```
int a, b, c;
a=2;        b=4;
c = a + b
```

  - 클래스 프로그래밍(C++, C#, Java 등) - 실제 생활 관련 프로그래밍
    - 자동차가 이동하고, 자동차가 정지하는 프로그램 작성 하려면 ??
      - 명사인 자동차 type (class) 필요
      - 기존 type(데이터 형) 으로 프로그래밍 불가능
    - 손님이 사과장수에게 사과를 구입하는 것을 프로그램으로 작성 하려면 ??
      - 명사인 손님 type , 사과장수 type, 사과 type 필요
  - 새로운 데이터형(type) 필요 → 기존 type 과 다른 type 필요 → 클래스

6

Class, 객체 ?

- 나는 사과장수에게 사과 2개를 샀다 → (대)명사는 변수(객체)
  - 어떤 type 의 변수가 필요 ?? → 기존 type으로 불가능 → 새로운 type 필요
  - FruitSeller, FruitBuyer 등의 type(클래스)는 제공 없음 → 본인이 만들어야 함.

```
FruitSeller a;
FruitBuyer b;
Apple c;
… …
```

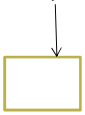
Type → class

변수 → 객체

객체지향

- 클래스(class) → C 에서 데이터 형(type)
  - 제공하는 것 사용 혹은 새로 작성
  - 속성 → 멤버변수
  - 행위 → 멤버함수
- 객체(object) / 인스턴스(instance) : C / C++ 에서 변수에 해당
  - 객체는 클래스에서 정의한 멤버변수, 멤버변수를 가짐

```
//type 변수;
int x;
```



```
// 클래스 객체;
Car my_car;
```

변수(인스턴스 변수)

함수(메소드)

8

객체지향...

- 다음과 같은 실제 세계의 객체에서 가능한 상태와 동작은 ?

객체	상태, 속성(멤버변수)	동작, 행위(멤버함수)
라디오	볼륨, 채널	볼륨을 높이다.
강아지	나이, 이름	짖다, 사료를 먹다
통장	잔액	입금하다, 출금하다, 이체하다.

9

(참고) C 구조체

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

struct account {
        int year;
        float money;
};

void main() {
        struct account me;          // 구조체 account 선언 필요
        me.year = 1987;
        me.money = 12.5;
        cout << me.year << me.money << endl;
}
```

객체지향 클래스

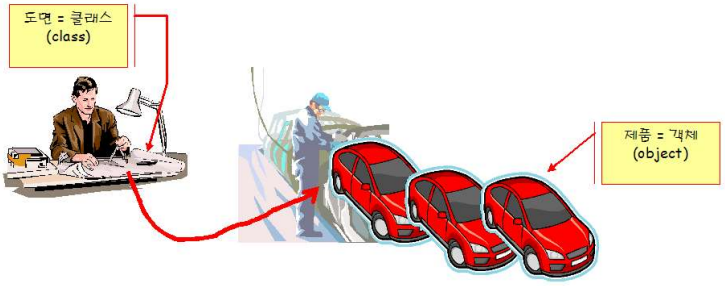
```
#include <iostream>
using namespace std;

class Account  {   // class (데이터 형) → 개발자가 작성 or 기존것 사용.
public :
     float money;    // 잔액

     void Input(float m ){ // 입금하다
            money += m;
     }
} ;

void main( ){

     Account me, father;    // 객체(변수), Account type(클래스) 선언 필요

     me.Input(12.5);

     father.Input(20.5);
}
```

11

클래스

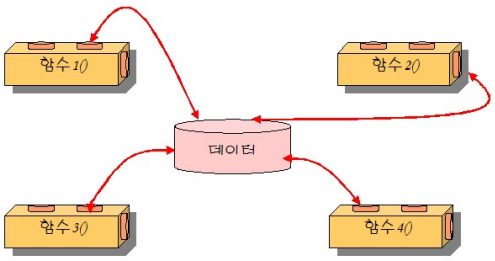
- 클래스(class): 객체를 만드는 설계도 → C 에서는 type
  - c 의 구조체 선언과 비슷
- 클래스로부터 만들어지는 것 → 객체, 인스턴스(instance).

```
int x;          → C 방식 변수 선언(타입명 변수명)
Account me;     → C++ 객체 선언(클래스명 객체명)
Car x, y, z;
```



절차 지향과 객체 지향

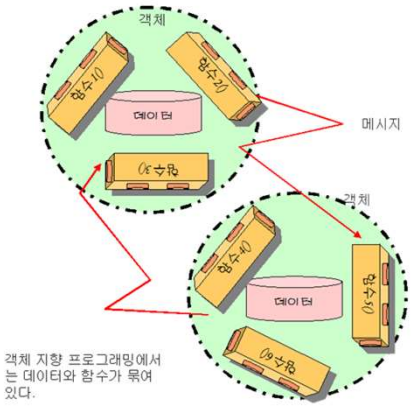
- 절차 지향 프로그래밍(Procedural Programming)
  - 문제를 해결하는 절차를 중요하게 생각하는 소프트웨어 개발 방법. 이들 절차는 모두 함수라는 단위로 묶이게 된다.



절차 지향 프로그래밍에서는 데이터와 함수가 묶여 있지 않다.

절차 지향과 객체 지향

- 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.

객체 지향의 개념들(나중에...)

- 캡슐화(encapsulation)
- 정보 은닉(information-hiding)
- 상속(inheritance)
- 다형성(polymorphism)

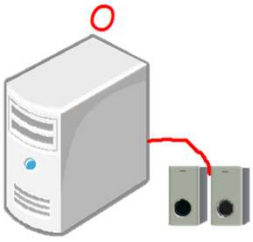
캡슐화

- 캡슐화(encapsulation)란 데이터(멤버변수)와 연산(멤버함수)들을 객체 안에 넣어서 묶는다는 의미이다.

정보 은닉

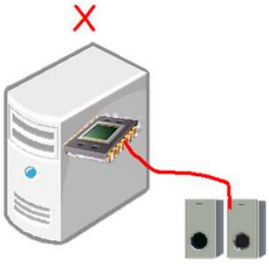
- 객체 내부의 데이터와 구현의 세부 사항을 외부 세계에게 감추는 것.
- 외부 세계에 영향을 끼치지 않으면서 쉽게 객체 내부를 업그레이드 할 수 있다.

O



만약 외부의 표준 오디오 단자를 이용하였으면 내부의 사운드 카드를 변경할 수 있다.

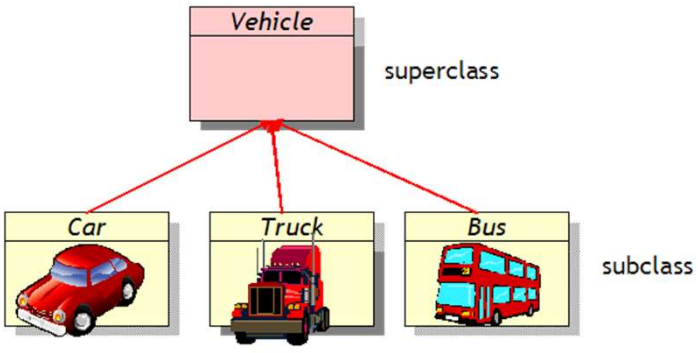
X



만약 내부의 오디오 제어칩의 단자에 연결하였으면 내부의 사운드 카드를 변경할 수 없다.

상속

- 상속은 기존의 코드를 재활용하기 위한 기법으로 이미 작성된 클래스(부모 클래스)를 이어받아서 새로운 클래스(자식 클래스)를 생성하는 기법이다.

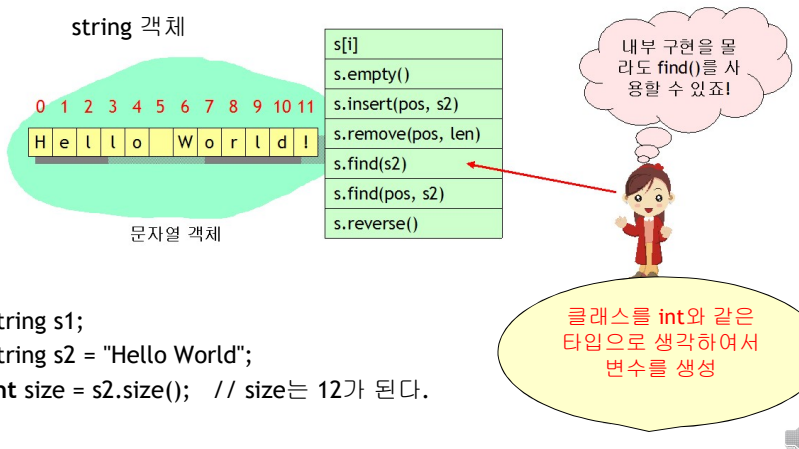


다형성

- 다형성이란 객체가 취하는 동작이 상황에 따라서 달라지는 것을 의미한다. → 함수 이름의 재사용

std::string 클래스

- C 에서 문자열은 문자형 배열/포인터 사용
- C++에서는 문자열을 나타내는 클래스 string을 제공한다.
  - 제공되는 클래스들은 내부 구조 알 필요 없이 사용 방법만 알면 됨




```
string s1;
string s2 = "Hello World";
int size = s2.size();   // size는 12가 된다.
```

객체 생성의 예

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

중요) string 은 std 공간에 정의되어 있음

```
int main(){
    string s1 = "This is a test.";   // string 객체를 생성하고 초기화한다.
    string s2;                       // 비어있는 string 객체를 생성한다.

    cout << s1 << endl;
    cout << "문자열을 입력하시오: " << endl;
    cin >> s2;
    cout << s2 << endl;
    return 0;
}
```

```
This is a test.
문자열을 입력하시오: This
This
```

s1	s2
멤버변수 = "This …"	멤버변수
멤버함수들 empty() insert(…) remove(…) find(…) find(…)…	멤버함수들 empty() insert(…) remove(…) find(…) find(…)…

string 클래스의 멤버 함수

string s = "Hello World";

멤버 변수/함수	설명
s[i]	i번째 원소 // 배열처럼 사용
s.empty()	s가 비어있으면 true 반환
s.insert(pos, s2)	s의 pos 위치에 문자열 s2를 삽입, 위치는 0 부터 시작
s.remove(pos, len)	s의 pos(index) 위치에서 len만큼을 삭제
s.find(s2)	s에서 문자열 s2가 발견되는 첫번째 인덱스를 반환, 없으면 -1 (string::npos) 반환
s.find(pos, s2)	s의 pos 위치부터 문자열 s2가 발견되는 첫번째 인덱스를 반환

멤버 함수 호출의 예

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

int main(){
// string 객체를 생성하고 초기화한다.
    string s1 = "This is a test.";
    string s2 = "This is a test.";

    s1.insert(4, "Hello");
    cout << s1 << endl;

    int index = s1.find("test");
    cout << index << endl;

    s1.append("World");
    cout << s1 << endl;

    s2.insert(2, "Hello");
    cout << s2 << endl;

    index = s2.find("is");
    cout << index << endl;

    s2.append("World");
    cout << s2 << endl;

    return 0;
}
```

```
ThisHello is a test.
15
ThisHello is a test.World

ThHellois is a test.
7
ThHellois is a test.World
```

문자열의 결합 / 비교

- 문자열 초기화 방법 1, 문자열 결합

```
string s1 = "Hello";
string s2 = " World";
string sentence = s1 + s2;  // "Hello World"
```

- 문자열 초기화 방법 2 , 문자열 비교 방법

```
string s1("Hello"), s2(" World");
if( s1 == s2 )
    cout << "동일한 문자열입니다" << endl;
else
    cout << "동일한 문자열이 아닙니다" << endl;
```

문자열 결합

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

int main()
{
    string s1("Slow"), s2("steady");
    string s3 = "the race.";
    string s4;

    s4 = s1 + " and " + s2 + " wins " + s3;
    cout << s4 << endl;
    return 0;
}

// 뒤의 예제 소스 생략
```

```
Slow and steady wins the race.
계속하려면 아무 키나 누르십시오 . . .
```

string 클래스 입출력

- 한 줄 전체 읽기 <-- “this is a book” 입력(enter)

```
string s;

cin >> s;    // this is a book 입력 → 입력에서 빈칸 전 까지만 저장
cout << s;  // this

getline(cin, s);        // this is a book 입력 → 한 줄씩 읽기
cout << s;              // this is a book

cout << s[3];           // s
```

- 한 문자씩 읽기

```
char c;
do {
    cin.get(c);         cout << c;
    …
} while( c != ‘\n’);
```

Report

- 154 쪽~
- 8번, 9번, 13번

- 제출 파일 : OOP4_학번.txt

```
void main(){
    p8( );    // 8번
    P9();     // 9번
    P13();    // 13번
}
```

Report

- 8번 힌트
  - 사용자 입력은 getline() 사용 string 객체(변수)에 저장
  - 알파벳 찾는 방법
    - if ((text[i] >= 'a' && text[i] <= 'z') || (text[i] >= 'A' && text[i] <= 'Z'))
  - 숫자 찾는 방법
    - if (text[i] >= '1' && text[i] <= '9')
  - 빈칸 찾는 방법
    - if (text[i] == ' ')

Report

- 9번 힌트
  - 사용자 입력은 getline() 사용 한 줄을 읽고 string 객체(변수)에 저장
  - 단어의 갯수 → 빈칸 갯수 ??

- 13번 힌트
  - string 클래스의 replace( ) 멤버함수 이용 → 사용 방법 인터넷서 찾아볼 것.

Q & A