상속

상속(inheritance)란?

- **객체 지향 프로그래밍**을 구성하는 중요한 특징
- 클래스간 **계층적 관계**를 구성
- 클래스의 **모든 멤버 변수와 멤버 함수를 물려받아** 새로운 클래스를 작성
- 기존 정의된 클래스를 **베이스 클래스(base class)**, **부모 클래스(parent class)**, **상위 클래스(super class)**라 한다
- 상속에 의해 새롭게 작성된 클래스를 **파생 클래스(derived class)**, **자식 클래스(child class)**, **하위 클래스(sub class)**라 한다

접근 제어 지시자

- public:
  - 어디서든 접근 허용

- protected:
  - 직접적인 상속 관계의 클래스 내부까지만 접근 허용

- private:
  - 해당 클래스 내부에서만 접근 허용

상속(inheritance)

```
class 부모클래스명
{
    …
};
class 파생클래스명 : 접근제어지시자 부모클래스명
{
    …
};
```

- **접근 제어 지시자를 생략**하면 파생 클래스의 **접근 제어 권한**은 **private**로 기본 **설정**된다
- **쉼표(,)**를 사용하여 상속 받을 **부모 클래스를 여러 개 명시할 수 있다**
- 상속 받는 **부모 클래스가 하나**이면 **단일 상속(single inheritance)**라 하고 **여러 개의 부모 클래스**를 상속 받으면 **다중 상속(multiple inheritance)**라 한다

상속(inheritance)

```
class Person
{
private:
    std::string _name;
    unsigned int _age;
};

class Student : public Person
{
private:
    std::string _class;
    std::string _grade;
};

class Worker : public Person
{
private:
    std::string _department;
    std::string _rank;
    unsigned int _pay;
};
```

멤버 이니셜라이저(Member Initializer)

- 초기화 대상을 명확히 인식할 수 있다
- 선언과 동시에 초기화가 이루어지기 때문에 성능에 이점이 있다
- 상수(const) 멤버 변수의 초기화가 가능하다
- 반드시 사용해야 될 경우
  - 상수 멤버가 있을 때
  - 레퍼런스 멤버가 있을 때
  - 멤버의 생성자를 호출해야 할 때
  - 상속받은 클래스일 경우 부모 생성자를 호출해야 할 때

```
생성자() : 멤버변수명(값), …
{
    …
}
```

멤버 이니셜라이저(Member Initializer)

```
생성자() : 멤버변수명(값), …
{
    …
}

자식생성자() : 부모생성자(값, …), …
{
    …
}
```