정리노트 #6 가상 함수와 추상 클래스

13조 (전대원, 소재현)

예제 9-1

```
 #include <iostream>
using namespace std;
class Base {
public:
void f() { cout << "Base::f() called" << endl; }
};
class Derived : public Base {
public:
void f() { cout << "Derived::f() called" << endl; }
};
void main() {
Derived d, *pDer;
pDer = &d;
pDer->f(); // Derived::f() 호출
Base* pBase;
pBase = pDer; // 업캐스팅
pBase->f(); // Base::f() 호출
}
```

함수가 재정의 되어 Derived::f() called

Base::f() called 결과 도출됨

가상 함수와 오버라이딩 구분하기

가상 함수

ㅇ virtual 키워드로 선언된 멤버 함수

ㅇ virtual 키워드의 의미

동적 바인딩 지시어로 사용

컴파일러에게 함수에 대한 호출 바인딩을 실행 시간까지 미루도록 한다

오버라이딩

파생 클래스에서 기본 클래스의 가상 함수와 동일한 이름의 함수 선

언

ㅇ 기본 클래스의 가상 함수의 존재감 상실시킴

ㅇ 파생 클래스에서 오버라이딩한 함수가 호출되도록 동적 바인딩

ㅇ 함수 재정의라고도 부름

ㅇ 다형성의 한 종류

```
class Base {
public:
virtual void f() {
cout << "Base::f() called" << endl;
}
};
```

```
class Derived : public Base {
public:
virtual void f() {
cout << "Derived::f() called" << endl;
}
};
```

이 부분에서 virtual void f() { 를 가상함수로 사용

virtual void f() { 를 오버라이딩으로 사용

함수 재정의

```
class Base {
public:
void f() {
cout << "Base::f() called" << endl;
}
};
class Derived : public Base {
public:
void f() {
cout << "Derived::f() called" << endl;
}
```

};

함수 재정의와 가상함수, 오버라이딩과 구분하였다

동적 바인딩

기본 클래스에 대한 포인터로 가상 함수를 호출하는 경우

ㅇ 객체 내에 오버라이딩한 파생 클래스의 함수를 찾아 실행

ㅇ 실행시간 바인딩, 런타임 바인딩, 늦은 바인딩으로 불림

오버라이딩 특징과 주의사항

오버라이딩 시 virtual 지시어 생략 가능

ㅇ 가상 함수의 virtual 지시어는 상속됨, 파생 클래스에서 virtual 생략 가능

ㅇ 가상 함수의 접근 지정

ㅇ private, protected, public 중 자유롭게 지정 가능

소멸자

가상 소멸자

ㅇ 소멸자를 virtual 키워드로 선언

ㅇ 소멸자 호출 시 동적 바인딩 발생

가상 소멸자의 경우(코드)

```
class Base {
public:
virtual ~Base();
};
```

```
class Derived: public Base {
public:
virtual ~Derived();
int main() {
Base *p = new Derived();
delete p;
}
};
```

delete p; 에서 ~Base() 소멸자 호출하였음

virtual ~Base(); >>> virtual ~Derived(); ~Derived() 실행 하였음

virtual ~Derived(); >>> virtual ~Base(); ~Base() 실행 하였음

추상 클래스

추상 클래스 : 최소한 하나의 순수 가상 함수를 가진 클래스이다

```
class Shape { // Shape은 추상 클래스
Shape *next;
public:
void paint() {
draw();
}
virtual void draw() = 0; // 순수 가상 함수
};
void Shape::paint() {
draw(); // 순수 가상 함수라도 호출은 할 수 있다.
}
```

추상 클래스 특징

- 추상 클래스의 포인터는 선언 가능
- 온전한 클래스가 아니므로 객체 생성 불가능
- 추상 클래스를 단순 상속하면 자동 추상 클래스
- 추상 클래스를 상속받아 순수 가상 함수를 오버라이딩