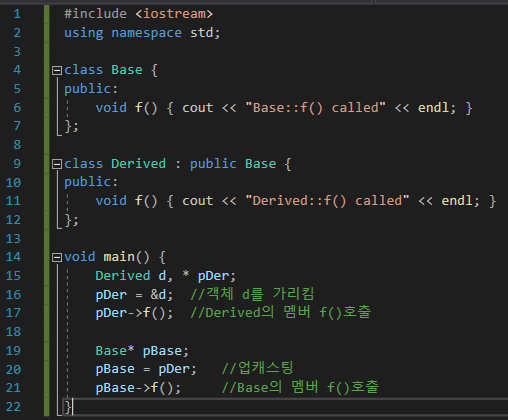
9.1 상속관계에서의 함수 재정의

함수재정의 기본클래스의 멤버함수와 동일한 이름과 원형으로 함수를 파생클래스에 다시 정의하는 것

예제9-1



객체 기본클래스의 함수, 재정의된 파생클래스의 함수를 가짐

ex) Derived d; //Base의 f(), Derived의 f()를 모두 가짐

파생클래스 포인터할당, 포인터접근 호출 ex) pDer = &d; //객체 d를 가리킴

pDer->f(); //Derived의 멤버 f()호출

기본클래스 포인터 업캐스팅, 포인터 접근 호출 ex) Base\* pBase;

pBase = pDer; //업캐스팅

pBase->f(); //Base의 멤버 f()호출

이런 호출관계는 컴파일시 결정 🡪 정적 바인딩

범위지정연산자 :: 기본클래스의 멤버함수, 파생클래스에 재정의된 함수 구분해 호출가능

ex) pDer->f(); //Derived의 멤버 f()호출

pDer->Base::f(); //Base의 멤버 f()호출

9.2 가상함수와 오버라이딩

가상함수, 오버라이딩 상속에 기반

객체지향언어의 메인

가상함수와 오버라이딩 오버라이딩의 개념 파생클래스에서 기본클래스에 작성된 가상함수 재작성

기본클래스에 작성된 가상함수 무력화

기본클래스 포인터에 접근해 호출하든, 파생클래스의 포인터에 접근해 호출하든 파생클래스의 오버라이딩된 가상함수가 실행됨

가상함수선언과 오버라이딩 가상함수 virtual 키워드로 선언된 멤버함수

기본클래스, 파생클래스 어디에서나 선언 가능

함수 오버라이딩 = 오버라이딩

파생클래스에서 기본클래스의 가상함수를 재정의하는 것

함수 재정의 컴파일시간 다형성

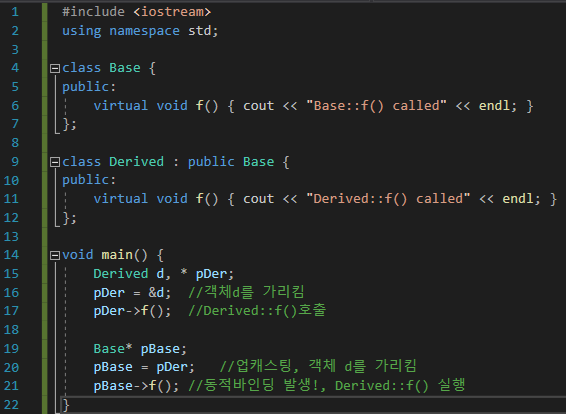
오버라이딩 실행시간 다형성

변수 오버라이딩 멤버함수에만 오버라이딩 가능 🡪 변수 오버라이딩이란 용어는 없음

함수재정의와 오버라이딩 함수재정의 – 정적바인딩(컴파일시간에 결정된 함수 호출)

오버라이딩 – 동적바인딩(함수가 호출되는 런타임에 동적바인딩)

오버라이딩과 가상함수 호출 오버라이딩 사례 예제9-2



객체 d에는 2개의 f()함수가 있음 but Derived의 f()가 Base의 f()를 무시하도록 오버라이딩됨

오버라이딩의 목적 파생클래스들이 자신의 목적에 맞게 가상함수를 재정의하는 것

기본클래스의 가상함수 상속받는 파생클래스에서 구현해야할 일종의 함수인터페이스 제공

오버라이딩을 통한 다형성 ex) class Shape {

public:

virtual void draw() {}

};

class Circle :public Shape {

protected:

virtual void draw() {}

};

class Rect :public Shape {

protected:

virtual void draw() {}

};

void paint(Shape\* p) {

p->draw();

}

int main()

{

paint(new Circle());//shape의 포인터가 circle로 업캐스팅, circle의 draw 실행

paint(new Rect());//shape의 포인터가 Rect로 업캐스팅, Rect의 draw 실행

}

동적바인딩 동적바인딩:오버라이딩된 함수가 무조건 호출 가상함수를 호출하는 코드 컴파일 🡪 컴파일러는 바인딩을 실행시간에 결정하도록 미룸

가상함수가 호출되면, 실행중에 객체 내에 오버라이딩된 가상함수를 동적으로 찾아 호출

오버라이딩 파생클래스에서 재정의한 가상함수의 호출 보장

동적바인딩이 발생하는 구체적 경우 파생클래스의 객체에 대해, 기본클래스의 포인터로 가상함수가 호출될 때 동적바인딩 발생

1. 기본클래스 내의 멤버함수가 가상함수 호출
2. 파생클래스
3. main()과 같은 외부함수에서 기본클래스의 포인터로 가상함수 호출
4. 다른 클래스에서 가상함수 호출

동적바인딩 사례 ex) class Shape {

public:

virtual void paint() {

draw();

}

virtual void draw() { cout << "Shape::draw() called" << endl; }

};

class Circle :public Shape {

protected:

virtual void draw() { cout << "Circle::draw() called" << endl; }

};

int main()

{

Shape\* pShape = new Circle();

pShape->paint();

delete pShape;

}

override 지시어 컴파일러가 오버라이딩을 확인하도록 하는 지시어

ex) class Circle :public Shape {

protected:

void draw() override { cout << "Circle::draw() called" << endl; }

};

오버라이딩이 아닐된 함수가 아닐 시, 컴파일 오류 발생

final 지시어 파생클래스에서 오버라이딩 금지, 상속금지

사용 경우 1. 가상함수의 원형 뒤에 final지시어 사용 🡪 파생클래스는 오버라이딩 할 수 없음

ex) class Shape {

public:

virtual void paint() {

draw();

}

virtual void draw() final { cout << "Shape::draw() called" << endl; }

}

2. 클래스 이름 뒤에 final 지시어 사용 🡪 이 클래스는 당할 수 없음

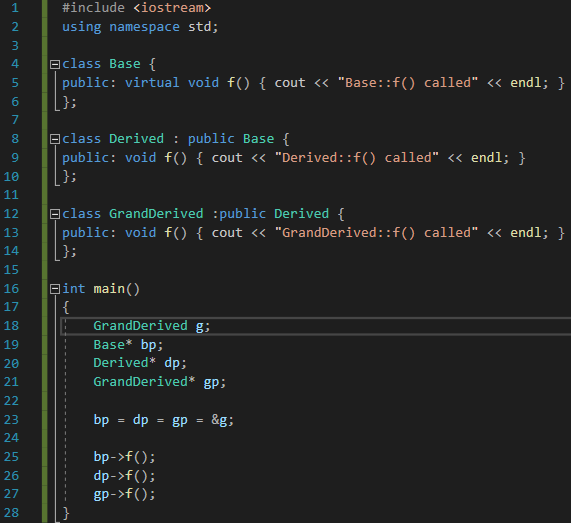
ex) class Shape final{ };

C++ 오버라이딩의 특징 오버라이딩의 성공조건과 실패 가상함수의 이름, 매개변수의 타입∙개수, 리턴타입이 일치해야 함

오버라이딩 시 virtual 지시어 생략가능 virtual 속성은 상속되는 성질이 있음 🡪 파생클래스에서 virtual 키워드를 생략해도 자동으로 가상함수가 됨

가상함수의 접근지정 가상함수에도 접근지정자 사용 가능

예제9-3



오버라이딩과 범위지정연산자(::) 오버라이딩당한 기본클래스의 가상함수 호출

main함수에서 기본클래스의 가상함수 호출 ex) Circle circle;

Shape\* pShape = &circle;

pShape->Shape::draw();

클래스의 멤버함수에서 기본클래스의 가상함수 호출 실제 응용에서 많이 이용

기본클래스의 가상함수를 그대로 활용하고 기능을 추가

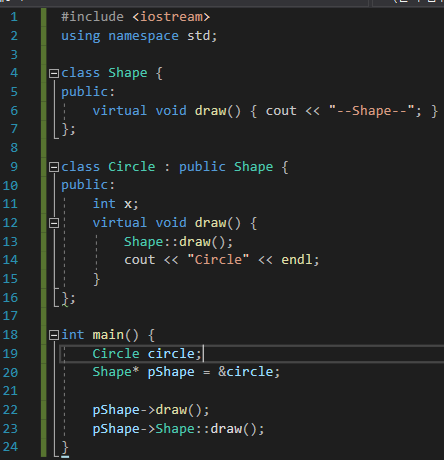
ex) class Circle :public Shape {

public:

virtual void draw() { Shape::draw(); }

};

예제9-4



범위지정연산자 범위규칙 동일한 이름의 변수, 함수가 여러곳에 선언되어 있을 때, 가장 가까운 범위에 선언된 이름을 사용하는 규칙

전역변수, 지역변수 이름이 같은 경우 ex) int n = 11;

int main() {

int n = 3;

cout << ::n << endl;

cout << n << endl;

}

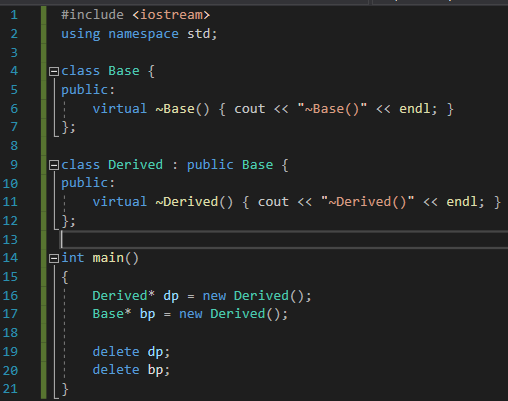
가상소멸자 소멸자를 만들 때 가상함수로 만들 것 🡪 파생클래스의 객체가 기본클래스의 대한 포인터로 delete되는 상황에 정상적인 소멸이 되도록하기 위함

소멸자를 가상함수로 선언하지 않은 경우 기본클래스타입의 포인터를 업캐스팅하고 delete하는 경우, 컴파일러는 기본클래스의 소멸자만 실행∙파생클래스의 소멸자는 실행X

소멸자를 가상함수로 선언한 경우 기본클래스의 소멸자가 파생클래스의 소멸자로 동적바인딩됨

파생클래스의 소멸자 실행 후 기본클래스의 소멸자 실행

예제 9-5



오버로딩, 함수재정의, 오버라이딩 비교 오버로딩 매개변수타입∙개수가 다른 함수들을 여러 개 중복 작성

오버라이딩 기본클래스의 가상함수를 파생클래스에서 재작성해, 동적바인딩을 유발시킴

함수재정의 가상함수가 아닌 멤버함수를 재작성해 정적 바인딩으로 처리하는 것

check time 1. (1) 2 B::g()

1. (2) ~A(), ~B()

1. (3) ~B(), ~A(), ~B(), ~A()

9.3 가상함수와 오버라이딩의 활용 사례

가상함수 파생클래스에서 오버라이딩할 함수를 알려주는 인터페이스

가상함수를 가진 기본클래스의 목적 기본클래스에서 가상함수 선언 🡪 파생클래스에서 오버라이딩, 각자의 함수를 선언하도록 유도

가상함수 오버라이딩: 파생클래스마다 다르게 구현하는 다형성 오버라이딩한 함수가 호출되는 것을 보장받음

동적바인딩 실행: 파생클래스의 가상함수 실행 동적바인딩을 이용해 기본클래스 포인터가 가리키는 파생클래스 객체를 바꿔가며 오버라이딩한 함수 호출

기본클래스의 포인터 활용 기본클래스의 포인터로 동적바인딩을 통해 파생클래스의 오버라이딩한 함수 실행

9.4 추상 클래스

순수가상함수 기본클래스에 작성된 가상함수 대부분 파생클래스에서 재정의해 구현할 함수를 알려주는 인터페이스의 역할 수행 🡪 선언O, 코드X

순수가상함수 함수의 코드가 없고 선언만 있는 가상함수

형식 virtual 반환형 함수이름() =0;

ex) virtual void draw() = 0;

추상클래스 하나 이상의 순수가상함수를 가진 클래스

= 불완전한 클래스 추상클래스로 인스턴스를 생성할 시 컴파일 오류 발생

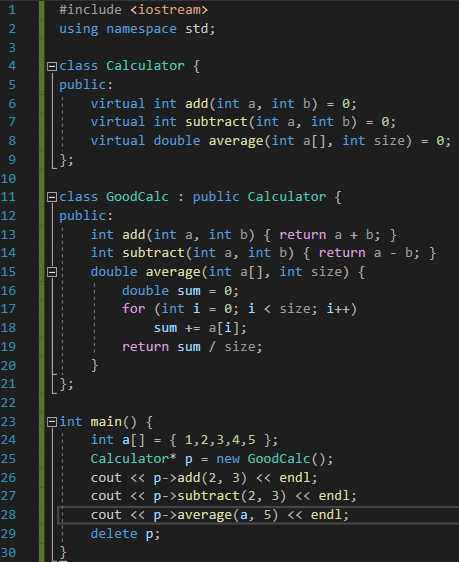
추상클래스의 목적 상속을 위한 기본클래스로 활용

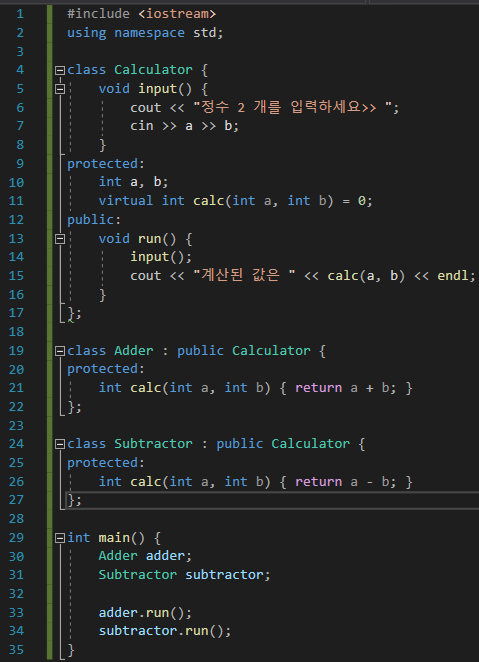
파생클래스가 구현할 함수의 원형을 보여주는 인터페이스 역할

추상클래스 상속 추상클래스를 상속받는 파생클래스는 자동으로 추상클래스가 됨

추상클래스의 구현 파생클래스에 추상클래스의 순수가상함수의 코드 작성

추상클래스의 모든 순수가상함수를 오버라이딩해야함

예제9-6 

예제9-7 

추상클래스의 용도 응용프로그램의 설계와 구현 분리

계층적 상속 관계를 가진 클래스의 구조를 만들 때 용이

check time 1. 3

2. 4