C++ 보고서

클래스의 상속에 대하여

로봇 게임단

19기 수습단원

이원준

목차

1. 서론

이 보고서는 c++ 프로그래밍 상속의 개념에 대해 설명하고 상속의 이점과 실제 코드에서의 활용 방안에 대해 설명합니다. 이 보고서의 목적은 C++의 상속 개념을 체계적으로 설명하고, 상속을 적절히 활용할 수 있는 역량을 육성하는 데에 있습니다.

2. C++ 상속의 기본 개념

- **상속의 정의**

상속은 기존 클래스를 기반으로 새로운 클래스를 정의하는 c++의 기능을 말합니다. 이를 통해 부모 클래스의 데이터와 함수 멤버를 자식 클래스에서 재사용하거나 확장 수정할 수 있습니다. 상속은 코드를 더 간결하게 작성하는 데 큰 도움을 줄 수 있습니다.

- 부모 클래스와 자식 클래스

부모 클래스는 상속의 원본이 되는 클래스입니다. 이에 반해 자식 클래스는 부모클래스를 상속받아 정의되는 클래스입니다. 자식 클래스는 부모 클래스의 멤버를 그대로 사용할 수 있으며, 추가적인 멤버를 정의하거나, 기본 클래스의 동작을 오버라이딩하여 변경할 수 있습니다

- 상속의 기본 문법 구조

C++에서 상속을 구현하는 기본 문법은 다음과 같습니다.

class 자식클래스이름 : 접근제한자 부모클래스이름 { }

여기서 접근제한자는 public, private, protected 중 하나를 사용할 수 있습니다. 접근 제한자는 부모 클래스의 변수와 함수가 자식 클래스에서 어느 정도로 접근 가능한지를 결정합니다. 예를 들어, public 상속을 사용할 경우 기본 클래스의 public 멤버는 파생 클래스에서도 public으로 유지되고, protected 멤버는 protected로 유지됩니다. 반면 private 멤버는 파생 클래스에서 직접 접근할 수 없습니다.

3. 상속의 유형 및 데이터 접근 관리

C++에서 상속은 기본 클래스의 멤버를 파생 클래스가 어떻게 접근할 수 있는지에 따라 접근 제한자를 기준, 세 가지 주요 유형으로 나눌 수 있습니다. 각각의 상속 유형은 접근 제어와 코드 재사용성을 다르게 처리하기 때문에 적절한 상황에 맞는 상속 유형을 선택하는 것이 중요합니다.

- 접근 제한자의 종류

1. public (공용)

정의 : public으로 선언된 멤버는 클래스 외부에서도 자유롭게 접근할 수 있습니다.

용도: 클래스 사용자에게 공개하고자 하는 멤버를 선언할 때 사옹됩니다.

특징: 객체나 포인터를 통해 어디서나 접근 가능하며, 파생 클래스에서도 접근 가능합니다.

2. Protected (보호)

정의: protected로 선언된 멤버는 클래스 외부에서는 접근할 수 없지만, 파생 클래스에서는 접근 가능합니다.

용도: 클래스 냅와 파생클래스에서만 접근이 필요하지만 외부에서의 데이터를 숨기고자 할 때 사용합니다.

특징: 파생클래스 내부에서만 접근할 수 있으며, 객체를 통해서는 접근할 수 없습니다.

3. Private (비공개)

정의: private로 선언된 멤버는 클래스 외부 및 파생 클래스에서 모두 접근할 수 없습니다. 오직 클래스 내부에서만 접근이 가능합니다.

용도: 클래스 내부에서만 관리되어야 하는 멤버를 선언할 때 사용합니다.

특징: 객체나 파생 클래스에서는 접근할 수 없으며, 오직 해당 클래스 내부에서만 사용됩니다

아래 자료를 참고해서3가지 접근제한자를 바탕으로 나눈 상속의 유형에 대해 알아봅시

다.

- 각 상속 유형의 비교

상속 유형	부모 클래스 public	부모 클래스 protected	부모 클래스 private
공용 상속	public	protected	접근 x
보호 상속	protected	protected	접근 x
비공개 상속	private	private	접근 x

위 표를 통해 각 상속 유형이 기본 클래스의 멤버에 어떻게 접근하는지 한눈에 알 수 있습니다. 실무에서는 주로 **공용 상속**을 사용하며, 특정 상황에서 보호 또는 비공개 상속을 선택합니다. 상속 유형은 설계의 의도와 코드의 가독성, 재사용성에 큰 영향을 미치므로, 상황에 맞는 적절한 상속 방식을 선택하는 것이 중요합니다.

- 접근 권한 설정 키워드 (friend, static, const)

friend, static, const는 C++에서 다양한 상황에서 사용되는 키워드로, 각각 객체 지향 프로그래밍과 관련된 중요한 역할을 합니다. 이 키워드들은 클래스 설계에서 멤버의 접근 권한과 동작을 제어하는 데 큰 영향을 미칩니다. 각 키워드가 의미하는 바와 그 용도, 그리고 함께 사용되는 경우에 대해 설명하겠습니다.

1. 키워드 friend

friend 키워드는 특정 클래스나 함수가 다른 클래스의 private 또는 protected 멤버에 접할 수 있도록 허용할 때 사용됩니다. friend로 지정된 클래스나 함수는 원래 접근할 수 없는 멤버에 대해 예외적으로 접근 권한을 가지게 됩니다.

Friend 가 함수에서의 역할에 대해 설명해보겠습니다. friend함수는 일반 함수나 멤버 함수중 다른 클래스의 비공개 멤버에 접근 할 수 있는 것들을 정의할 때 사용할 수 있습니다.

Friend가 클래스에서의 역할에 대해 설명하겠습니다. 두 개의 클래스가 있다고 할 때, 하나의 클래스가 다른 클래스의 비공개 및 보호 멤버에 접근할 수 있도록 friend 관계를 설정할 수 있습

니다.

2. 키워드 static

static 키워드는 **정적 멤버**를 정의하는 데 사용됩니다. 클래스 멤버가 static으로 선언되면, 해당 멤버는 클래스의 모든 인스턴스에 대해 **공유**됩니다. 즉, 여러 객체가 동일한 static 멤버를 공유하며, 특정 객체에 종속되지 않습니다.

Static 멤버 변수에 대해 설명하겠습니다. 정적 멤버 변수는 클래스 내에서 선언되지만, 클래스 외부에서 초기화되어야 합니다. 모든 객체는 해당 변수를 공유하므로, 값이 변경되면 클래스의 모든 객체에서 해당 변경이 반영됩니다

Static 멤버 함수에 대해 설명하겠습니다. 정적 멤버 함수는 객체가 아니라 클래스 자체에 종속되므로 **객체 없이** 호출할 수 있습니다. 이러한 함수는 정적 멤버 변수만 접근할 수 있으며, 비정적 멤버 변수나 함수에는 접근할 수 없습니다.

3. 키워드 const

const는 변수가 **상수**로서 변경 불가능하도록 만드는 키워드입니다. const는 다양한 곳에 사용될 수 있으며, 변수, 함수 매개변수, 반환 값 등을 상수화하여 의도하지 않은 변경을 방지할 수 있습니다.

Const 멤버 함수에 대해 설명하겠습니다. 멤버 함수가 const로 선언되면, 그 함수는 **객체의 상태를 변경할 수 없다는 것을 의미**합니다. 이 함수는 객체 내의 멤버 변수를 수정할 수 없으며, 읽기 전용 기능을 제공할 때 주로 사용됩니다.

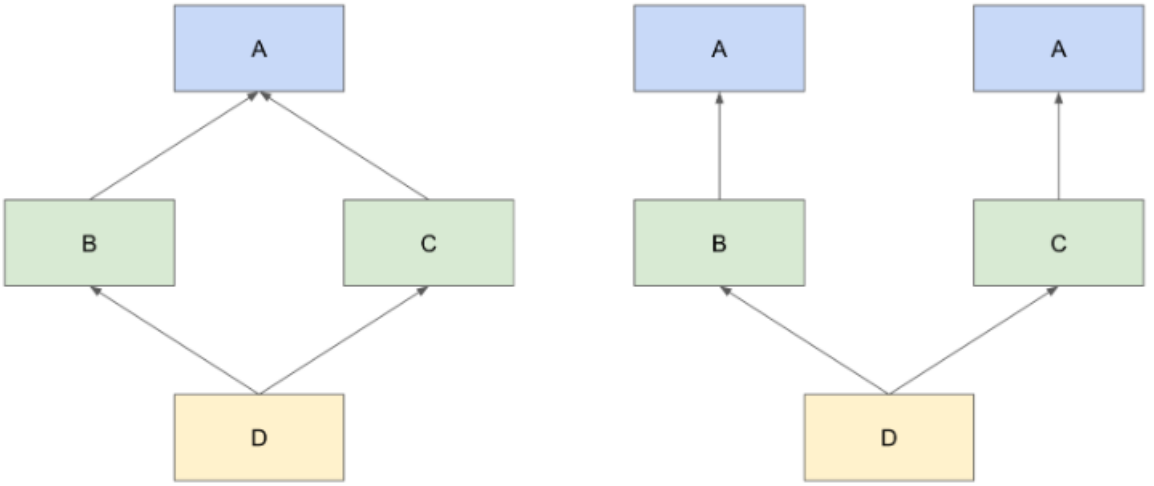
4. 가상 상속과 다중 상속

C++에서 **다중 상속**과 **가상 상속**은 복잡한 상속 구조에서 발생하는 문제를 해결하기 위한 중요한 개념입니다.

다중 상속

- 정의 : 다중 상속은한 클래스가 두 개 이상의 클래스로 부터 상속받는 것을 의미합니다. 즉, 하나의 클래스가 여러부모 클래스로부터 멤버 변수를 물려받아 사용할 수 있습니다.

- 문제점: 다이아몬드 문제가 발생할 수 있습니다. 이는 두 부모 클래스가 동일한 상위 클래스를 상속 받았을 때 생기는 모호성에 기인합니다. 다음 페이지 사진을 참고합시다.



[1] [2]

- 사진에서 1번과 2번은 다중 상속이 복수로 해석될 여지가 있음을 보여줍니다.
- 다이아몬드 문제의 해결방법은 다음 단락을 참고합시다.

가상 상속

- 정의: **가상 상속**은 **다이아몬드 문제**를 해결하기 위해 도입된 기능입니다. 부모 클래스에서 가상 상속을 지정하면, 상속받는 모든 자식 클래스가 **같은 상위 클래스의 인스턴스를 공유**합니다.

  가상 상속은 다중상속시 부모 클래스의 중복을 피하고, 상위 클래스의 단일 인스턴스만을 유지하게 합니다.

- 원리: 가상 상속의 원리

  가상 상속은 다중 상속에서 공통 부모 클래스가 있을 때, 파생 클래스에 공통된 상위 클래스의 단일 인스턴스만을 상속받도록 합니다.

  부모 클래스에서 가상 상속을 지정하면, 파생 클래스에서는 같은 상위 클래스의 인스턴스를 여러 번 가질 필요가 없으며, 이를 통해 메모리 낭비와 모호성을 방지합니다.

  - 가상 상속의 메모리 구조

    가상 상속에서는 파생 클래스마다 상위 클래스의 독립적인 인스턴스를 생성하지 않고, 포인터를 통해 단일 인스턴스를 공유합니다. 이는 메모리 관리 측면에서 효율적일 수 있지만, 성능 저하의 요인이 될 수 있습니다.
    .

정리

구분	다중 상속	가상 상속
정의	하나의 클래스가 여러 부모 클래스로부터 상속	공통된 부모 클래스의 단일 인스턴스만 상속
문제점	다이아몬드 문제로 모호성이 발생할 수 있음	모호성 문제 해결, 공통 부모의 단일 인스턴스 공유
구현 방식	일반적인 상속 구조	`virtual` 키워드를 사용하여 상속
메모리 사용	부모 클래스마다 인스턴스가 생성됨	부모 클래스의 단일 인스턴스만 생성

5. 추상 클래스와 순수 가상 함수

추상 클래스

추상 클래스는 하나 이상의 **순수 가상 함수**를 포함하는 클래스입니다. 객체를 직접 생성할 수

없고, 다른 클래스가 이를 상속받아 구체적인 구현을 해야만 사용될 수 있습니다. 주로 상속 계층 구조에서 공통된 인터페이스나 기능을 정의하는 데 사용됩니다.

- 추상 클래스 자체로는 객체를 만들 수 없다.
- 추상 클래스를 상속받은 자식 클래스에서 모든 순수 가상 하수를 구현해야 합니다.
- 공통된 매서드를 제공하거나, 자식 클래스가 반드시 구현해야 하는 매서드를 강제할 수 있습니다.

순수 가상 함수

순수 가상 함수는 추상 클래스 내에서 정의된 함수로, 해당 클래스에서는 구현되지 않고, 이를 상속받은 클래스에서 반드시 구현해야 합니다. C++에서는 순수 가상 함수는 함수 선언 끝에 = 0으로 표현합니다.

- 추상 클래스를 만들기 위한 핵심 요소입니다.
- 함수의 선언만 존재하며, 구현은 자식 클래스에서 이루어져야 합니다.
- 순수 가상함수를 클래스는 추상 클래스가 됩니다

추상 클래스와 순수 가상 함수의 활용

- 공통 인터페이스 제공: 추상 클래스는 여러 파생 클래스에 공통된 메소드를 제공합니다.

6. 결론

7. 참고 문헌