C++ 퀴즈 (2023-01-05)

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 다음 코드의 출력 값과 그 이유는?

Base 생성자, derived 생성자, base 소멸자

Derived를 할당하므로 base, derived 순서대로 생성자가 호출됨

Base 포인터로 업캐스팅한 것이고, 소멸자가 가상함수가 아니므로 base 소멸자만 호출

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 다음 코드의 출력 값과 그 이유는?

Base 생성자, 2

Base를 할당했으므로 Base생성자 호출

pb를 형변환 하더라도 pb는 Base 객체이므로 Base의 정보를 가지고 있고 Derived의 정보를 가지고 있지 않음

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 다음 코드의 출력 값과 그 이유는?

*(reinterpret\_cast는 관련 없는 포인터끼리의 강제적인 형 변환을 가능하게 한다. )*

65537

Visual studio x64 기준 short는 2 바이트, int는 4 바이트

Class A 는 [0x0001][0x0001] Class B 는 [0x00000001]

따라서 A를 B로 강제 형변환 하면 int에 0x00010001이 들어감

따라서 16^4 \* 1 + 16^0 \* 1 = 65537

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 다음 코드의 문제점에 대해 서술하시오.

Class B 는 int a 와 int b 를 멤버로 가지게 됨

따라서 a를 담는 배열에 넣으면 크기가 맞지 않아 내용이 잘리는 문제 발생

이는 배열을 객체 포인터 배열이 아닌 객체 배열로 만들어 배열의 내부 크기가

한정되는 것이 근본적인 원인이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 다시 한번 이 문제에 대해 생각해보자.

블로그에선 직원 데이터와 매니저 데이터를 관리하기 위해 이중 포인터를 선언하였고, 이를 new를 통해 동적 할당하여 Employee\*의 배열과 Manager\*의 배열을 생성하였다. 그리고 이 배열을 다시 동적 할당하여 동적 할당된 지역에 Employee와 Manager를 선언하였는데, 반드시 이럴 필요가 있을지 생각해 보자.

다시 말하자면, Employee\*만 선언하여 먼저 배열을 동적 할당하고, 여기에 Employee를 대입하는 방법으로 블로그에서 구현했던 기능을 완전히 같게 구현할 수 있지 않을까?

Employee 가 상속하는 멤버 함수를 가상함수로 만들어 Employee 포인터로 Manager 클래스를 업캐스팅 하더라도 Manager의 멤버함수를 사용할 수 있게끔 하면 된다.