(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2015-0067580
(43) 공개일자 2015년06월18일

(51) 국제특허분류(Int. Cl.)
G06F 21/57 (2013.01) *G06F 9/45* (2006.01)
(21) 출원번호 10-2013-0153191
(22) 출원일자 2013년12월10일
심사청구일자 2013년12월10일

(71) 출원인
충남대학교산학협력단
대전광역시 유성구 대학로 99 (궁동, 충남대학교)
(72) 발명자
류재철
대전 유성구 어은로 57, 132동 801호 (어은동, 한빛아파트)
조은선
대전 유성구 어은로 57, 110동 1504호 (어은동, 한빛아파트)
(뒷면에 계속)
(74) 대리인
권혁수, 송윤호

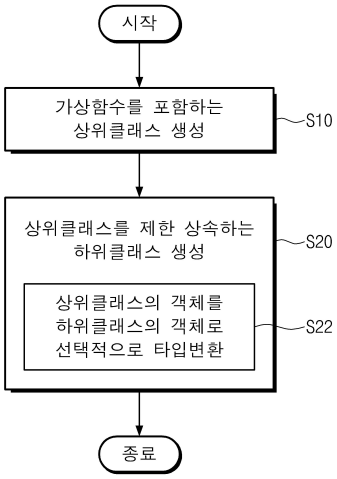
전체 청구항 수 : 총 11 항

(54) 발명의 명칭 **가상 테이블의 취약성을 완화시키는 타입 변환 방법**

(57) 요 약

본 발명은 적어도 하나의 가상 함수를 포함하는 상위 클래스 및 상위 클래스 구조체를 제한 상속하는 하위 클래스를 포함하고, 상위 클래스의 객체를 하위 클래스의 객체로 타입 변환하는 것을 제한적으로 허용하는 타입 변환 방법에 관한 것이다.

대 표 도 - 도4



(72) 발명자 **최중현**
세종 금남면 진동길 65-3,

윤태섭
충청남도 예산군 대흥면 예당관광로 443-1

이 발명을 지원한 국가연구개발사업
과제고유번호 13-912-06-003
부처명 미래창조과학부
연구관리전문기관 충남대학교 산학협력단
연구사업명 방송통신원천기술개발사업
연구과제명 안드로이드 신규 취약점 탐지를 위한 모바일 소프트웨어 보안 테스팅 도구 개발
기 여 율 1/1
주관기관 한국방송통신전파진흥원(KCA)
연구기간 2013.04.01 ~ 2016.03.31

명 세 서

청구범위

청구항 1

적어도 하나의 가상 함수를 포함하는 상위 클래스를 생성하는 단계; 및

상기 상위 클래스를 제한 상속하는 하위 클래스를 생성하는 단계;

를 포함하고,

상기 하위 클래스를 생성하는 단계는 상기 상위 클래스의 객체를 상기 하위 클래스의 객체로 선택적으로 타입 변환하는 단계;

를 포함하는 타입 변환 방법.

청구항 2

제1항에 있어서,

상기 제한 상속은 상기 하위 클래스를 선언할 때 제1키워드를 추가하여 수행되는 타입 변환 방법.

청구항 3

제2항에 있어서,

상기 제1키워드는 weak 인 타입 변환 방법.

청구항 4

제1항에 있어서,

상기 선택적으로 타입 변환하는 단계는

제2키워드에 의해 정의되는 타입 변환 함수에 의해서 수행되고,

상기 제2키워드에 의해 정의되는 타입 변환 함수는 상기 제1키워드에 의해 선언된 클래스의 객체의 타입 변환을 허용하지 않는 타입 변환 방법.

청구항 5

제4항에 있어서,

상기 제2키워드는 wstatic_cast 인 타입 변환 방법.

청구항 6

제4항에 있어서,

상기 제2키워드에 의해 정의되는 타입 변환 함수는 제3키워드에 의해 선언된 클래스의 객체의 타입 변환은 허용하는 타입 변환 방법.

청구항 7

제6항에 있어서,

상기 제3키워드는 nweak 인 타입 변환 방법.

청구항 8

제6항에 있어서,

상기 하위 클래스는 상기 상위 클래스에 포함된 가상 함수와는 다른 가상 함수를 더 포함하고,

상기 제3키워드에 의해 선언된 클래스의 객체는 상기 다른 가상 함수를 호출할 수 없는 타입 변환 방법.

청구항 9

제6항에 있어서,

상기 제3키워드에 의해 선언된 클래스는 상기 상위 클래스에 포함된 가상 함수를 재정의할 수 있는 타입 변환 방법.

청구항 10

제6항에 있어서,

상기 제3키워드에 의해 선언된 포인터는 일반 포인터로 타입 변환되지 않는 타입 변환 방법.

청구항 11

CPU와 메모리, 입력 장치 및 출력 장치를 포함하는 컴퓨터 장치에 있어서,

상기 CPU는 적어도 하나의 가상 함수를 포함하는 상위 클래스를 생성하고, 상기 상위 클래스를 제한 상속하는 하위 클래스를 생성하되,

상기 하위 클래스를 생성함에 있어서 상기 상위 클래스의 객체를 상기 하위 클래스의 객체로 선택적으로 타입 변환하는 것을 특징으로 하는 컴퓨터 장치.

발명의 설명

기 술 분 야

[0001] 본 발명은 가상 테이블의 취약성을 완화시키는 타입 변환 방법에 관한 것이다.

배 경 기 술

[0002] 프로그래밍 언어 중에서 C나 C++은 성능을 최우선적으로 고려하여 구현이 되어서 상대적으로 구조화나 추상화가 덜 되어 있다. 이로 인해, C나 C++에서는 여러 다른 언어에 비해서 취약성 탐지가 중요하다.

[0003] C나 C++에는 타입 방법과 관련된 취약성이 존재한다. 특히 C++에서는 상위 클래스 타입을 하위 클래스 타입으로 변환할 수 있는 타입 변환을 지원하는데, 상위 클래스 타입을 하위 클래스 타입으로 변환할 때 동적인 타입 검사를 지원하지 않기 때문에 다음과 같은 취약성을 지니고 있다.

[0004] 도 1 내지 도 3은 기존의 타입 변환 방법의 취약성을 설명하기 도면이다. 도 1을 참조하면, 상위 클래스 A가 선언되고, 상위 클래스 A를 상속한 하위 클래스 B가 선언되어 있다. 이 때, 도 2에 도시된 코드를 실행하면 Line 1에서 a는 A 클래스로 선언되고, Line 2에서 A클래스형 포인터 변수로 p가 선언되면서 a의 주소가 p에 대입된다. Line 3에서 p를 정적 타입 변환 하여 B 클래스형 포인터 변수에 대입하게 되면, 에러 없이 타입 변환

이 수행된다. 하지만, p는 원래 a를 가리키므로 Line 4에서 f2()를 호출하게 되면 A 클래스는 f2() 함수를 정의하고 있지 않기 때문에 문제가 발생한다. C++에서 다형성을 지원하기 위한 가상 함수는 도 3에 도시된 것처럼 각 클래스마다 가상 테이블로 관리되는데, 각 객체는 자신에게 적합한 가상 테이블에 대한 포인터를 객체 내에 내포하고 있고, 이로부터 함수 시작 주소를 판단한다. 따라서, 도 2의 Line 4에서 f2()를 호출하게 되면 A의 가상 테이블 범위 밖의 엉뚱한 주소를 함수 시작 주소로 인식하기 때문에 런타임에 예기치 않은 오류가 발생할 수 있다. 그리고, 이를 이용하여 악성 코드의 취약점 공격(exploit)에 이용될 수 있는 문제점이 있다. 이를 타입 혼동(Type Confusion)이라 한다.

발명의 내용

해결하려는 과제

[0005] 본 발명의 목적은 C++에서 가상 테이블의 취약성을 완화시키는 타입 변환 방법을 제공하는 데 있다.

[0006] 본 발명의 기술적 과제는 이상에서 언급한 기술적 과제들로 제한되지 않으며, 언급되지 않은 또 다른 기술적 과제들은 아래의 기재로부터 당업자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0007] 본 발명의 실시예에 따른 타입 변환 방법은 적어도 하나의 가상 함수를 포함하는 상위 클래스를 생성하는 단계 및 상기 상위 클래스를 제한 상속하는 하위 클래스를 생성하는 단계를 포함하고, 상기 하위 클래스를 생성하는 단계는 상기 상위 클래스의 객체를 상기 하위 클래스의 객체로 선택적으로 타입 변환하는 단계를 포함할 수 있다.

[0008] 실시예에서, 상기 제한 상속은 상기 하위 클래스를 선언할 때 제1키워드를 추가하여 수행될 수 있다.

[0009] 실시예에서, 상기 제1키워드는 weak 일 수 있다.

[0010] 실시예에서, 상기 선택적으로 타입 변환하는 단계는 제2키워드에 의해 정의되는 타입 변환 함수에 의해서 수행되고, 상기 제2키워드에 의해 정의되는 타입 변환 함수는 상기 제1키워드에 의해 선언된 클래스의 객체의 타입 변환을 허용하지 않을 수 있다.

[0011] 실시예에서, 상기 제2키워드는 wstatic_cast 일 수 있다.

[0012] 실시예에서, 상기 제2키워드에 의해 정의되는 타입 변환 함수는 제3키워드에 의해 선언된 클래스의 객체의 타입 변환은 허용할 수 있다.

[0013] 실시예에서, 상기 제3키워드는 nweak 일 수 있다.

[0014] 실시예에서, 상기 하위 클래스는 상기 상위 클래스에 포함된 가상 함수와는 다른 가상 함수를 더 포함하고, 상기 제3키워드에 의해 선언된 클래스의 객체는 상기 다른 가상 함수를 호출할 수 없을 수 있다.

[0015] 실시예에서, 상기 제3키워드에 의해 선언된 클래스는 상기 상위 클래스에 포함된 가상 함수를 재정의할 수 있다.

[0016] 실시예에서, 상기 제3키워드에 의해 선언된 포인터는 일반 포인터로 타입 변환되지 않을 수 있다.

[0017] 본 발명의 실시예에 따른 컴퓨터 장치는 CPU와 메모리, 입력 장치 및 출력 장치를 포함하고, 상기 CPU는 적어도 하나의 가상 함수를 포함하는 상위 클래스를 생성하고, 상기 상위 클래스를 제한 상속하는 하위 클래스를 생성하되, 상기 하위 클래스를 생성함에 있어서 상기 상위 클래스의 객체를 상기 하위 클래스의 객체로 선택적으로 타입 변환할 수 있다.

발명의 효과

[0018] 본 발명의 일 측면에 따르면, C++에서 가상 테이블의 취약성을 완화시킬 수 있는 타입 변환 방법을 제공할 수 있다.

[0019] 본 발명의 효과가 상술한 효과들로 한정되는 것은 아니며, 언급되지 아니한 효과들은 본 명세서 및 첨부된 도면들로부터 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 명확히 이해될 수 있을 것이다.

도면의 간단한 설명

[0020] 도 1 내지 도 3은 C++에서 기존의 타입 변환 방법이 가지는 취약성을 설명하기 위한 도면이다.

도 4는 본 발명의 실시예에 따른 타입 변환 방법을 설명하기 위한 도면이다.

도 5는 본 발명의 실시예에 따른 타입 변환 방법에서 하위 클래스가 상위 클래스를 제한 상속하는 것을 설명하기 위한 도면이다.

도 6은 본 발명의 실시예에 따른 타입 변환 방법에서 제한 상속에 의한 타입 변환 제한을 설명하기 위한 도면이다.

도 7 및 도 8은 본 발명의 실시예에 따른 "nweak" 키워드를 설명하기 위한 도면이다.

도 9는

발명을 실시하기 위한 구체적인 내용

[0021] 본 발명의 다른 이점 및 특징, 그리고 그것들을 달성하는 방법은 첨부되는 도면과 함께 상세하게 후술 되는 실시 예를 참조하면 명확해질 것이다. 그러나 본 발명은 이하에서 개시되는 실시 예에 한정되는 것이 아니라 서로 다른 다양한 형태로 구현될 수 있으며, 단지 본 실시 예는 본 발명의 개시가 완전하도록 하고, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 발명의 범주를 완전하게 알려주기 위해 제공되는 것이며, 본 발명은 청구항의 범주에 의해 정의될 뿐이다.

[0022] 만일 정의되지 않더라도, 여기서 사용되는 모든 용어들(기술 혹은 과학 용어들을 포함)은 이 발명이 속한 종래 기술에서 보편적 기술에 의해 일반적으로 수용되는 것과 동일한 의미를 가진다. 일반적인 사전들에 의해 정의된 용어들은 관련된 기술 그리고/혹은 본 출원의 본문에 의미하는 것과 동일한 의미를 갖는 것으로 해석될 수 있고, 그리고 여기서 명확하게 정의된 표현이 아니더라도 개념화되거나 혹은 과도하게 형식적으로 해석되지 않을 것이다.

[0023] 본 명세서에서 사용된 용어는 실시 예들을 설명하기 위한 것이며 본 발명을 제한하고자 하는 것은 아니다. 본 명세서에서, 단수형은 문구에서 특별히 언급하지 않는 한 복수형도 포함한다. 명세서에서 사용되는 '포함한다' 및/또는 이 동사의 다양한 활용형들 예를 들어, '포함', '포함하는', '포함하고', '포함하며' 등은 언급된 조성, 성분, 구성요소, 단계, 동작 및/또는 소자는 하나 이상의 다른 조성, 성분, 구성요소, 단계, 동작 및/또는 소자의 존재 또는 추가를 배제하지 않는다.

[0024] 본 명세서에서 '및/또는' 이라는 용어는 나열된 구성들 각각 또는 이들의 다양한 조합을 가리킨다.

[0025] 본 발명은 C++에서 다형성 지원을 위해 가상 함수를 사용함으로써 발생하는 취약성을 완화시키는 타입 변환 방법에 관한 것이다.

[0026] 본 발명의 실시예에 따르면, 제한 상속을 위한 새로운 상속 키워드와 제한 상속이 이루어진 경우에 타입 변환을 제한하는 새로운 타입 변환 키워드가 정의될 수 있다. 실시예에 따르면, 제한 상속을 위한 새로운 상속 키워드는 "weak"로 정의될 수 있고, 타입 변환을 제한하는 새로운 타입 변환 키워드는 "wstatic_cast"로 정의될 수 있다. 상기 "weak"와 "wstatic_cast"는 예시적인 것으로 필요에 따라 변형될 수 있다.

[0027] 본 발명의 실시예에 따르면, "weak" 키워드에 의해 제한 상속된 하위 클래스는 "wstatic_cast" 키워드를 이용하여 타입 변환을 할 수 있다. 그러나, 기존의 C++ 타입 변환 방법과는 달리 "wstatic_cast" 키워드를 이용하여 상위 클래스의 객체를 하위 클래스의 객체로 타입 변환하는 경우에는 컴파일러가 컴파일 단계에서 이를 검출할 수 있다. 따라서, 배경 기술에서 지적한 타입 혼동과 같은 문제를 방지할 수 있다. 이하 첨부된 도면을 참조하여, 보다 상세히 설명할 것이다.

[0028] 도 4는 본 발명의 실시예에 따른 타입 변환 방법을 설명하기 위한 도면이다.

[0029] 도 4에 도시된 것과 같이, 본 발명의 실시예에 따른 타입 변환 방법은 적어도 하나의 가상 함수를 포함하는 상위 클래스를 생성하는 단계(S10) 및 상위 클래스를 제한 상속하는 하위 클래스를 생성하는 단계(S20)를 포함할

수 있다. 상위 클래스를 제한 상속하는 하위 클래스를 생성하는 단계(S20)는 상위 클래스의 객체를 하위 클래스의 객체로 선택적으로 타입 변환하는 단계(S22)를 포함할 수 있다.

[0030] 본 발명의 실시예에 따른 타입 변환 방법은 C++을 이용하여 사용자가 프로그래밍한 코드를 컴파일할 때, 컴퓨터 장치의 중앙 처리 장치(CPU)에 의해 수행될 수 있다. 상위 클래스를 제한 상속하는 하위 클래스를 생성하는 단계(S20)에 대해서는 도 5를 참조하여 보다 상세히 설명할 것이며, 상위 클래스의 객체를 하위 클래스의 객체로 선택적으로 타입 변환하는 단계(S22)에 대해서는 도 6 내지 도 8을 참조하여 보다 상세히 설명할 것이다.

[0031] 도 5는 본 발명의 실시예에 따른 타입 변환 방법에서 상위 클래스를 제한 상속하는 하위 클래스를 생성하는 단계(S20)를 설명하기 위한 도면이다.

[0032] 도 5를 참조하면, Line 1 내지 Line 5에서 상위 클래스인 A가 정의되어 있다. 상위 클래스인 A는 가상 함수인 f1() 함수를 포함하고 있다. 상위 클래스에서 "virtual" 키워드를 이용하여 함수를 가상으로 선언하게 되면, A 클래스를 상속하는 하위 클래스들은 가상 함수를 오버라이딩(overriding)하여 사용할 수 있고, 실행시에 오버라이딩된 함수를 우선적으로 호출하게 된다.

[0033] Line 6 내지 Line 10에서, 상위 클래스 A를 제한 상속하는 하위 클래스 B1이 정의되어 있다. 상술한 것처럼, "weak" 키워드를 이용하여 상속함으로써, 제한 상속으로 정의할 수 있다. Line 8에서는 상위 클래스인 A에서 정의된 f1()함수를 오버라이딩하고 있고, Line 9에서는 새로운 가상 함수를 정의하고 있다.

[0034] Line 11 내지 Line 14에서, 상위 클래스 A를 기존의 방법으로 상속하는 하위 클래스 B2가 정의되어 있다. 마찬가지로 Line 12에서 f1()함수를 오버라이딩하고 있다.

[0035] 도 6은 본 발명의 실시예에 따른 타입 변환 방법에서 제한 상속에 의한 타입 변환 제한을 설명하기 위한 도면이다.

[0036] 도 6을 참조하면, 도 2에서 설명한 것과 동일하게 Line 1에서 a는 A 클래스로 선언되고, Line 2에서 p는 A 클래스형 포인터 변수로 선언되면서 a의 주소가 p에 대입된다. 그러나, 도 2와 달리 Line 3 및 Line 4에서 "wstatic_cast" 키워드를 이용하여 타입 변환이 수행된다.

[0037] 앞서 설명한 것처럼, "wstatic_cast" 키워드를 이용하여 상위 클래스 객체를 "weak" 키워드를 이용하여 제한 상속한 하위 클래스의 객체로 변환하는 경우, 컴파일러는 이를 검출하여 사용자에게 오류를 통지할 수 있다. 즉, Line 3에서 상위 클래스 A의 객체인 p를 "weak" 키워드를 이용하여 제한 상속한 하위 클래스인 B1의 객체로 "wstatic_cast"를 이용하여 타입 변환하려고 하기 때문에, 컴파일러는 사용자에게 오류 통지를 하게 된다. 이를 통해서, 가상 테이블 사용으로 인한 취약성을 완화시킬 수 있다.

[0038] 이와 달리, Line 4에서 하위 클래스 B2는 "weak" 키워드를 이용하여 제한 상속하지 않았기 때문에 정상적으로 타입 변환이 수행될 수 있다.

[0039] 본 발명의 실시예에 따르면, "weak" 키워드와 "wstatic_cast" 키워드를 보다 효율적으로 활용하기 위해서 "nweak" 키워드가 추가적으로 정의될 수 있다. "nweak" 키워드 역시 예시적인 것이며, 다른 키워드로 변형될 수 있다. 도 4와 도 6 내지 도 7을 참조하여 "nweak" 키워드에 대해서 설명할 것이다.

[0040] 도 7 및 도 8은 본 발명의 실시예에 따른 "nweak" 키워드를 설명하기 위한 도면이다.

[0041] 도 7을 참조하면, Line 1에서 b는 B1 클래스로 선언되고, Line 2에서 p는 A 클래스형 포인터 변수로 선언되면서 b의 주소가 p에 대입된다. 이 경우에는 p가 원래 B1 클래스의 객체이기 때문에 타입 변환이 수행되더라도 문제가 없으나, "wstatic_cast"를 이용한 타입 변환에서는 이러한 타입 변환을 오류로 인식하게 된다. 상기 문제점을 해결하기 위해 "nweak" 키워드가 이용될 수 있다.

[0042] 본 발명의 실시예에 따르면, "nweak" 키워드는 변수에 사용될 수 있다. 도 6의 Line 4에서 보여지는 것처럼, r은 "nweak" 타입의 B1 클래스로 선언될 수 있다. "nweak" 키워드를 통해 선언된 r은 "wstatic_cast"에 의해

타입 변환이 제한되지 않는다.

[0043] 본 발명의 실시예에 따르면, "nweak" 키워드에 의해 선언된 하위 클래스 객체는 상위 클래스에서 정의된 가상 함수는 호출할 수 있으나, 하위 클래스에서 추가된 가상 함수는 호출할 수 없다. 즉, 도 4를 참조하면, f1() 함수의 경우는 상위 클래스에 정의되어 있던 함수이기 때문에 "r->f1();"은 정상적으로 컴파일된다. 하지만 f2() 함수의 경우에는 상위 클래스에 정의되어 있지 않고, 하위 클래스인 B1에서 정의된 함수이기 때문에 "r->f2();"는 컴파일 단계에서 오류가 발생하게 된다. 이렇게 상위 클래스에서 정의되지 않고, 하위 클래스에서 추가된 가상 함수를 호출하지 못하게 하여 안정성을 더 강화할 수 있다.

[0044] 도 8은 본 발명의 실시예에 따른 "nweak" 키워드로 선언된 포인터와 다른 포인터 간의 관계를 설명하기 위한 도면이다.

[0045] 본 발명의 실시예에 따르면, "nweak" 키워드로 선언된 포인터는 다른 포인터로 변환되지 않지만, 다른 포인터는 "nweak" 키워드로 선언된 포인터로 지정될 수 있다.

[0046] 도 8을 참조하면, Line 1에서 nwp는 "nweak" B1 클래스형 포인터 변수로 선언되고, Line 2에서 p는 B1 클래스형 포인터 변수로 선언된다. Line 3에서 보는 바와 같이 B1 클래스형 포인터 변수를 "nweak" B1 클래스형 포인터 변수로 변환하는 것은 가능하다. 따라서, Line 4에서 nwp가 f1() 함수를 호출하여도 문제가 되지 않는다. 하지만, Line 5에서 보는 바와 같이 "nweak" B1 클래스형 포인터 변수를 일반 B1 클래스형 포인터 변수로 지정하는 것은 허용되지 않는다. p에 nwp가 지정되면 도 6에서 설명한 것처럼 하위 클래스에서 추가된 가상 함수를 호출할 수 없어야 하는데 p에 nwp가 지정되는 것이 허용되면 p가 f2() 함수를 호출할 수 있게 되기 때문에 이러한 포인터 변환은 허용되지 않는다.

[0047] 도 9는 본 발명의 실시예에 따른 타입 변환 방법을 수행하는 컴퓨터 장치를 나타낸 블록도이다. 도 4에 도시된 타입 변환 방법은 도 9의 컴퓨터 장치에서 수행될 수 있다.

[0048] 본 발명의 실시예에 따른 컴퓨터 장치는 CPU(10), 메모리(20), 입력부(30) 및 출력부(40)를 포함한다. CPU(10)는 메모리(20)로부터 읽어들인 자료를 이용하여 데이터 처리를 하며, 메모리(20)는 데이터를 저장할 수 있다. 입력부(30)는 사용자로부터 데이터를 입력받을 수 있고, 출력부(40)는 사용자에게 CPU(10)의 처리 결과를 제공할 수 있다.

[0049] 본 발명의 실시예에 따른 CPU(10)는 C++에 의해 작성된 프로그래밍 코드를 컴파일할 수 있다. CPU(10)는 컴파일 단계에서 적어도 하나의 가상 함수를 포함하는 상위 클래스를 생성하고, 상기 상위 클래스를 제한 상속하는 하위 클래스를 생성하되, 상기 하위 클래스를 생성함에 있어서 상기 상위 클래스의 객체를 상기 하위 클래스의 객체로 선택적으로 타입 변환할 수 있다.

[0050] 전술한 본 발명의 실시예에 따른 타입 변환 방법은 컴퓨터에서 실행되기 위한 프로그램으로 제작되어 컴퓨터가 읽을 수 있는 기록 매체에 저장될 수 있다. 상기 컴퓨터가 읽을 수 있는 기록 매체는 컴퓨터 방법에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 저장 장치를 포함한다. 컴퓨터가 읽을 수 있는 기록 매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광 데이터 저장 장치 등이 있다.

[0051] 이상의 실시예들은 본 발명의 이해를 돕기 위하여 제시된 것으로, 본 발명의 범위를 제한하지 않으며, 이로부터 다양한 변형 가능한 실시예들도 본 발명의 범위에 속할 수 있음을 이해하여야 한다. 예를 들어, 본 발명의 실시예에 도시된 각 구성 요소는 분산되어 실시될 수도 있으며, 반대로 여러 개로 분산된 구성 요소들은 결합되어 실시될 수 있다. 따라서, 본 발명의 기술적 보호범위는 특허청구범위의 기술적 사상에 의해 정해져야 할 것이며, 본 발명의 기술적 보호범위는 특허청구범위의 문언적 기재 그 자체로 한정되는 것이 아니라 실질적으로는 기술적 가치가 균등한 범주의 발명에 대하여까지 미치는 것임을 이해하여야 한다.

부호의 설명

[0052] 100: 컴퓨터 장치

10: CPU

20: 메모리

30: 입력부

40: 출력부

도면

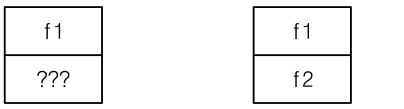
도면1

```
line
   1  class A{
   2        int x;
   3        public : A() {..}
   4        virtual void f1(){..}
   5  }
   6   class B : public A {
   7        public : B(){...}
   8        virtual void f1() {...}
   9        virtual void f2(){..}
  10  }
```

도면2

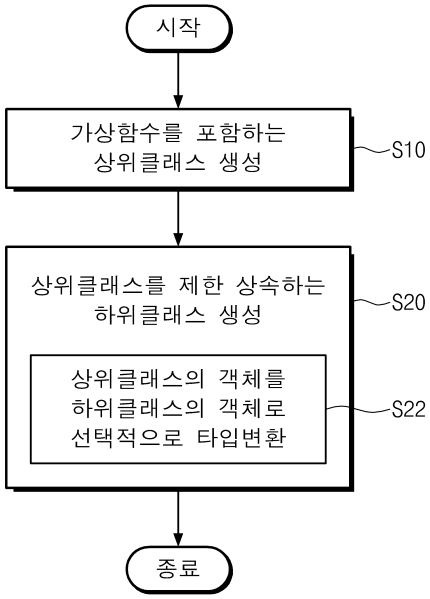
```
line
   1  A a;
   2  A* p = &a;
   3  B* q= static_cast<B*> p;
   4  q->f2();
```

도면3



(A의 가상 테이블) (B의 가상 테이블)

도면4



도면5

```
line
   1  class A{
   2        int x;
   3        public : A() {..}
   4        virtual void f1(){..}
   5  }
   6   class B1 : weak public A {
   7        public : B1(){...}
   8        virtual void f1() {...}
   9        virtual void f2(){..}
  10  }

  11  class B2: public A{
  12        public : B2() {...}
  13        virtual void f1() {...}
  13  }
```

도면6

```
line
   1  A a;
   2  A* p = &a;
   3  B1* q1= wstatic_cast<B1*> p;
   4  B1* q2= wstatic_cast<B2*> p;
```

도면7

```
line
   1  B1 b;
   2  A* p = &b;
   3  B1* q= wstatic_cast<B1*> p;
   4  nweak B1* r= wstatic_cast<B1*> p;
```

도면8

```
line
   1  nweak B1* nwp =..
   2  b1 * p =..

   3  nwp = p;
   4  nwp->f1();

   5  p = nwp;
```

도면9