

ICE2004 자료구조 실습 1

제 목

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2021년 10월 10일

학부 정보통신공학과

학년 2학년

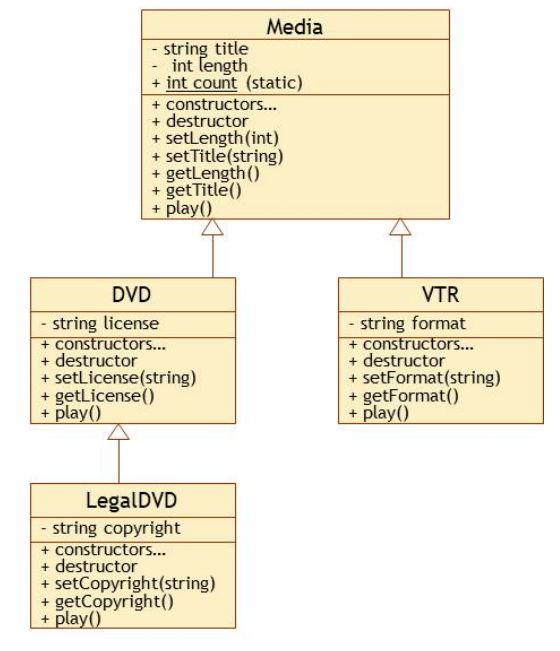
성명 김다영

학번 12201856



1. **개요**

* Media, VTR, DVD, LegalDVD 네 개의 class가 존재한다.
* DVD와 VTR은 Media를, LegalDVD는 DVD를 상속한다.
* 주어진 코드가 실행이 되도록 + 연산자와 << 연산자를 overloading 한다.
* Media 객체를 생성할 때마다 static int형인 count를 증가시켜 객체의 개수를 기록한다.
* 자식 객체의 함수의 overriding을 통해 다형성을 구현한다.
* 동적할당 된 객체의 메모리 해제를 위해 각 객체를 delete한다.
* main 함수에서 Media\* 형 객체를 만든 후에 자식에 해당하는 객체를 new를 이용해 동적 할당한다. 이때 자식에 대한 함수를 호출하기 위해선 동적바인딩을 이용해야 하며 그에 따라 virtual 선언을 해야 한다.



1. **코드 캡처 및 설명**

**❑ Media.h**

텍스트이(가) 표시된 사진

자동 생성된 설명

1. **멤버 함수의 virtual 선언**
   * Media\*형 객체에 동적할당 된 객체들의 함수를 호출하기 위해 Media의 소멸자와 play 함수에서 virtual을 선언해줘야 한다.
2. **함수의 오버로딩**
   * main에서 Media\*형에 대한 객체에 대해 +연산자와 <<연산자를 사용하기 위해 함수의 오버로딩을 선언해야 한다.
   * <<연산자의 경우 이미 정의된 ostream class 내부에 있는 함수를 변형하여 오버로딩 해야 하므로 friend를 이용해야 한다.
3. **static 변수**
   * 객체들의 수를 세기 위한 변수를 선언하기 위해 모든 클래스의 객체 내에서 공통으로 쓰일 수 있는 변수인 static 변수를 선언한다.

**❑ Media.cpp**

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

**① static 변수 초기화**

* + Static 멤버 변수는 모든 객체들이 접근해야 한다. 따라서 전역 범위에서 static 변수를 초기화한다.

**② 오버로딩 함수 정의**

- +연산자 오버로딩 함수의 정의 같은 경우 main 함수에서 +연산자에 의해 계산된 값이

mp[3]에 할당돼야 한다. 따라서 mp[3]의 자료형인 Media\*을 반환형으로 가져야 한다.

- +연산자 같은 경우 두 피연산자의 length를 더하는 역할을 한다. 하지만 반환형은

Media\*에 해당하므로 두 피연산자의 length를 더한 값을 length로 갖는 새로운

Media객체를 동적할당 하여 반환한다.

**❑ VTR.h**

텍스트이(가) 표시된 사진

자동 생성된 설명

**① 함수의 오버라이딩**

* + play 함수를 오버라이딩 하여 VTR 객체에서 play를 호출할 때 VTR의 play함수가 호출되도록 한다.

**❑ VTR.cpp**

텍스트이(가) 표시된 사진

자동 생성된 설명

**❑ DVD.h**

텍스트이(가) 표시된 사진

자동 생성된 설명

**① 함수의 오버라이딩**

* + play 함수를 오버라이딩 하여 DVD객체에서 play를 호출할 때 DVD의 play함수가 호출되도록 한다.

**❑ DVD.cpp**

텍스트이(가) 표시된 사진

자동 생성된 설명

**❑ LegalDVD.h**

텍스트, 모니터, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**① 함수의 오버라이딩**

* + play 함수를 오버라이딩 하여 LegalDVD 객체에서 play를 호출할 때 LegalDVD의 play함수가 호출되도록 한다.

**❑ LdgalDVD.cpp**

텍스트이(가) 표시된 사진

자동 생성된 설명

**❑main.cpp**

텍스트이(가) 표시된 사진

자동 생성된 설명

**① Media\*형 배열 선언**

* + Media\*형 배열을 선언하여 각각에 VTR, DVD, LegalDVD를 동적할당한다.
  + Media\*형으로 선언이 됐으므로 VTR, DVD, LegalDVD에 접근하기 위해서는 virtual 함수로 선언하여 동적바인딩이 가능하도록 해야한다.

**② static변수인 count의 증가**

* + VTR, DVD, LegalDVD의 각각의 생성자가 호출되기 전에 부모인 Media의 생성자가 호출된다. Media의 생성자에서는 static변수인 count를 객체가 생성될 때마다 1씩 증가시킨다. main 함수에서는 VTR을 2번, DVD와 LegalDVD를 각각 한 번씩 호출하므로 count가 4가 된다.
  + :: 연산자를 이용하여 Media 클래스 내의 static 변수인 count에 접근하여 for문의 범위를 설정한다.

**③ 동적바인딩**

* + Media\*형 배열 변수의 요소로 각 자식들의 객체가 동적할당 되었다. play함수는 virtual 선언이 되었으므로 포인터형 배열을 이용하여 play에 접근하면 Media에 대한 play 변수가 호출되는 것이 아닌 각 개체의 play 변수가 호출된다.

**④ 함수의 오버로딩**

* + <<연산자를 전역 함수로서의 오버로딩을 하여 두 개의 prameter를 갖는다. 각 배열의 요소를 cout하면 그 배열의 요소에 해당하는 객체가 play되고 있음을 출력하도록 했다. 따라서 cout<<mp[0]를 하면 mp[0]의 객체에 해당하는 VTR변수가 호출되고 있다는 문구가 출력된다.
  + +연산자를 Media 객체의 멤버 함수로서의 오버로딩을 하여 한 개의 prameter를 갖는다. +연산자에 대한 반환값이 mp[3]에 할당돼야 한다. mp[3]는 Media\*형 객체이므로 +연산자에 대한 반한형은 Media\*가 돼야한다. 함수 오버로딩 시 parameter로 들어가는 값은 연산자 오른쪽에 해당하는 변수의 자료형이다. 이때 연산자 오른쪽에 해당하는 자료형은 mp[2]이고 이는 Media\*형 변수에 해당하므로 parameter는 Media\*형이 돼야한다.

**⑤ 객체의 소멸자 호출**

* + 객체의 소멸자를 호출하기 위해 delete를 이용한다. main에 이용된 mp변수의 각 요소는 동적할당 된 경우이므로 소멸자 호출을 위해 delete를 꼭 사용해야 한다.
  + 위에서 생성한 객체의 수는 4개이므로 i가 0일때부터 4보다 작을 때까지 for loop가 실행되도록 한다.
  + mp 배열의 각 객체는 동적할당 되어 있고, Media의 소멸자가 virtual로 선언되어 있어 소멸자가 호출될 때 Media의 소멸자가 아닌, 각 객체에 해당하는 소멸자가 호출되게 된다.

1. **결과 캡처**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**① play 함수 호출 결과**

* mp는 Media\*형의 배열에 해당하며, mp의 각 요소들은 Media를 상속한 클래스의 객체가 동적할당 된 것이다. Media의 멤버 함수인 play를 정의할 때 virtual로 정의했으므로 배열의 각 요소에서 play 함수를 호출할 때 동적바인딩 되어 Media의 play가 호출되는 것이 아닌, 각 객체의 play가 호출된다.
* 따라서 출력 결과에서 “Media가 play되고 있습니다.”라는 문구가 출력되는 것이 아닌 각 객체가 play되고 있다는 문구가 출력이 되는 것을 볼 수 있다.

**② << 연산자 오버로딩 결과**

* << 연산자는 Media\*형에 해당하는 피연산자의 play함수를 호출하도록 오버로딩 되었다. 따라서 **cout<<mp[0];** 가 실행되면 mp[0]의 동적할당된 자료형의 play함수가 호출된다.
* mp[0]은 VTR 객체가 동적할당 된 것이므로 출력결과가 “VTR이 play되고 있습니다.”가 호출된다.

**③ + 연산자 오버로딩 결과**

* + 연산자는 두 개의 피연산자의 length를 더한 값을 length로 가지는 Media\*형 변수를 반환하도록 오버로딩 되어있다. 각 피연산자는 \*mp[1]과 mp[2]에 해당하고, mp[1]의 length는 3, mp[2]의 length는 10에 해당한다. 따라서 동적할당 되며 반환된 Media\*변수의 length는 13이 된다.
* 따라서 mp[3]의 getLength함수를 호출하면 13이 출력되는 것을 볼 수 있다.

**④ 소멸자 호출 결과**

* Media의 소멸자는 virtual 함수로 선언되었으며, Media\*형 배열인 mp의 각 요소는 Media를 상속한 객체가 동적할당 되어있다. 따라서 동적바인딩이 된 형태이므로 mp의 각 요소를 delete하게 되면 Media의 소멸자가 4번 호출되는 것이 아니라, 각 요소에 동적할당 된 객체들의 소멸자가 호출된다.
* 따라서 mp[0], mp[1], mp[2], mp[3]가 for loop를 통해 delete 됨에 따라 각각의 동적할당 된 자료형의 소멸자가 호출되는 것을 볼 수 있다.
* 동적할당 된 자료형의 소멸자만 호출되는 것이 아니라 상속받은 Media의 소멸자도 호출되는데 이는 소멸자가 호출될 때 자식이 소멸된 이후에 부모가 소멸이 돼야 하기 때문이다.