



C++ - 모듈 05

반복 및 예외

요약:

이 문서에는 C++ 모듈 중 모듈 05의 연습 문제가 포함되어 있습니다.

버전: 10

콘텐츠

I	소개	2
II	일반 규칙	3
III	연습 00: 엄마, 저 커서 관료가 되고 싶어요!	5
IV	연습 01: 자세를 가다듬어라, 구더기들아!	7
V	연습 02: 아니요, 28C가 아닌 28B 양식이 필요합니다...	9
VI	운동 03: 적어도 커피 만드는 것보다는 낫습니다.	11
VII	제출 및 동료 평가	13

1장 소개

C++는 비야른 스트로스트룹이 C 프로그래밍 언어 또는 '클래스가 있는 C'의 변형으로 만든 범용 프로그래밍 언어입니다(출처: [위키백과](#)).

이 모듈의 목표는 **객체 지향 프로그래밍**을 소개하는 것입니다. 이것이 C++ 여정의 시작점이 될 것입니다. OOP를 배우기 위해 많은 언어가 권장됩니다. C++는 여러분의 오랜 친구인 C에서 파생된 언어이기 때문에 C++를 선택하기로 결정했습니다. C++는 복잡한 언어이며, 코드를 단순하게 유지하기 위해 C++98 표준을 준수합니다.

최신 C++는 여러 측면에서 많이 다르다는 것을 알고 있습니다. 따라서 능숙한 C++ 개발자가 되고 싶다면 42개의 공통 코어를 넘어서는 것은 여러분에게 달려 있습니다!

2장 일반 규칙

컴파일

- c++와 -Wall -Wextra -Werror 플래그를 사용하여 코드를 컴파일합니다.
- 플래그 -std=c++98을 추가하면 코드가 계속 컴파일됩니다.

서식 및 이름 지정 규칙

- 연습 디렉터리의 이름은 ex00, ex01, ..., exn
- 가이드라인에서 요구하는 대로 파일, 클래스, 함수, 멤버 함수 및 속성의 이름을 지정하세요.
- 클래스 이름을 **대문자 대소문자** 형식으로 작성합니다. 클래스 코드가 포함된 파일은 항상 클래스 이름에 따라 이름이 지정됩니다. 예를 들어 `ClassName.hpp/ClassName.h`, `ClassName.cpp` 또는 `ClassName.hpp`. 예를 들어, 벽돌 벽을 의미하는 "BrickWall" 클래스의 정의가 포함된 헤더 파일이 있다면 그 이름은 `BrickWall.hpp`가 됩니다.
- 달리 지정하지 않는 한, 모든 출력 메시지는 새 줄 문자로 끝나야 하며 표준 출력에 표시되어야 합니다.
- *안녕, 노미네트!* C++ 모듈에는 코딩 스타일이 강제되지 않습니다. 좋아하는 스타일을 따를 수 있습니다. 하지만 동료 평가자가 이해할 수 없는 코드는 채점할 수 없는 코드라는 점을 명심하세요. 깔끔하고 읽기 쉬운 코드를 작성하기 위해 최선을 다하세요.

허용/금지

더 이상 C로 코딩하지 마세요. 이제 C++로 전환하세요! 그러므로:

- 표준 라이브러리의 거의 모든 것을 사용할 수 있습니다. 따라서 이미 알고 있는 것을 고수하는 대신 익숙한 C 함수의 C++ 버전을 최대한 많이 사용하는 것이 현명할 것입니다.
- 그러나 다른 외부 라이브러리는 사용할 수 없습니다. 즉, C++11(및 파생 형식) 및 Boost 라이브러리는 금지됩니다. 다음 함수도 금지됩니다: `*printf()`, `*alloc()` 및 `free()`. 이 함수를 사용하면 성적은 0점이 됩니다.

- 명시적으로 달리 명시되지 않는 한, 네임스페이스 <ns_name> 및 친구 키워드는 금지되어 있습니다. 그렇지 않으면 성적은 -42점이 됩니다.
- **모듈 08과 09에서만 STL을 사용할 수 있습니다.** 즉, 그때까지는 컨테이너(벡터/리스트/맵 등)와 알고리즘(<알고리즘> 헤더를 포함해야 하는 모든 것)을 사용할 수 없습니다. 그렇지 않으면 성적이 -42점이 됩니다.

몇 가지 설계 요구 사항

- 메모리 누수는 C++에서도 발생합니다. 메모리를 할당할 때(새로운 키워드)의 경우 **메모리 누수**를 방지해야 합니다.
- 모듈 02부터 모듈 09까지, **명시적으로 달리 명시된 경우를 제외하고** 클래스는 **정통 정석 양식**으로 디자인해야 합니다.
- 헤더 파일에 있는 모든 함수 구현(함수 템플릿 제외)은 연습에 0을 의미합니다.
- 각 헤더를 다른 헤더와 독립적으로 사용할 수 있어야 합니다. 따라서 필요한 모든 종속성을 포함해야 합니다. 그러나 **include 가드**를 추가하여 이중 포함 문제를 피해야 합니다. 그렇지 않으면 성적이 0점이 됩니다.

읽기

- 필요한 경우 추가 파일을 추가할 수 있습니다(예: 코드를 분할하는 경우). 이러한 과정은 프로그램에서 확인하지 않으므로 필수 파일을 제출하는 한 자유롭게 추가할 수 있습니다.
- 간혹 연습 지침이 짧아 보이지만 지침에 명시되지 않은 요구 사항이 예제에 나와 있는 경우가 있습니다.
- 시작하기 전에 각 모듈을 완전히 읽으세요! 정말 그렇게 하세요.
- 오딘의, 토르의! 머리를 써라!!!




많은 클래스를 구현해야 합니다. 자주 사용하는 텍스트 편집기를 스크립팅할 수 없다면 이 작업이 지루해 보일 수 있습니다.



운동을 완료하는 데는 어느 정도의 자유가 주어집니다. 그러나 필수 규칙을 따르고 게으르지 마세요. 유용한 정보를 많이 놓칠 수 있습니다! 이론적 개념에 대해 주저하지 말고 읽어보세요.

제3장

연습 00: 엄마, 저 커서 관료가 되고 싶어요!

	운동 : 00
엄마, 저 커서 관료가 되고 싶어요!	
제출 디렉토리 : <code>ex00/</code>	
제출할 파일 : <code>Makefile</code> , <code>main.cpp</code> , <code>Bureaucrat.{h, hpp}</code> , <code>Bureaucrat.cpp</code>	
금지된 기능 : 없음	



예외 클래스는 반드시 정통 정석 형식으로 디자인할 필요는 없습니다. 하지만 다른 모든 클래스는 반드시 그래야 합니다.

사무실, 복도, 양식, 대기열로 구성된 인공적인 악몽을 디자인해 봅시다.
재미있나요? 재미없나요? 안타깝네요.

먼저, 이 거대한 관료주의 기계에서 가장 작은 톱니바퀴인 **관료**부터 시작하세요. **관료**

에게는 반드시 있어야 합니다:

- 상수 이름입니다.
- 그리고 1(최고 등급)에서 150(최저 등급)까지의 등급이 부여됩니다.

유효하지 않은 등급으로 관료를 인스턴스화하려는 시도는 예외를 발생시켜야 합니다:
`Bureaucrat::GradeTooHighException` 또는 `Bureaucrat::GradeTooLowException`을 발생시킵니다.

이 두 가지 속성에 대한 getName() 및 getGrade() 함수를 제공합니다. 또한 관료 등급을 늘리거나 줄이는 두 개의 멤버 함수도 구현합니다. 등급이 범위를 벗어나면 두 함수 모두 생성자와 동일한 예외를 던집니다.



기억하세요. 1등급이 가장 높고 150등급이 가장 낮으므로 3등급을 올리면 관료에게 2등급이 주어져야 합니다.

던져진 예외는 시도 및 잡기 블록을 사용하여 잡을 수 있어야 합니다:

```
시도
{
    /* 역주: 관료들과 함께 일하기 */.
}
캐치 (STD::예외 & E)
{
    /* 핸들 예외 */
}
```


삽입(") 연산자의 오버로드를 구현하여 (꺾쇠 괄호 없이) 다음과 같은 내용을 인쇄합니다:

<이름>, 관료 등급 <등급>.

평소와 같이 몇 가지 테스트를 통해 모든 것이 예상대로 작동하는지 확인합니다.

제4장

연습 01: 자세를 가다듬어라, 구더기들아!

	운동 : 01
전열을 가다듬어라, 구더기들아!	
체크인 디렉토리 : <i>ex01/</i>	
제출할 파일 : 이전 연습의 파일 + Form.{h, hpp}, Form.cpp	
금지된 기능 : 없음	

이제 관료가 생겼으니 그들에게 할 일을 주자고요. 양식 더미를 작성하는 것보다 더 좋은 활동이 있을까요?

그런 다음 **Form** 클래스를 만들어 보겠습니다. 이미 만들었습니다:

- 상수 이름입니다.
- 서명 여부를 나타내는 부울입니다(작성 시에는 서명되지 않음).
- 서명하려면 일정한 성적이 필요합니다.
- 그리고 이를 실행하기 위해서는 일정한 성적이 필요합니다.

이러한 모든 속성은 비공개이며 보호되지 않습니다.

양식의 성적은 관료에게 적용되는 것과 동일한 규칙을 따릅니다. 따라서 양식 성적이 범위를 벗어나는 경우 다음과 같은 예외가 발생합니다:

Form::GradeTooHighException 및 Form::GradeTooLowException.

이전과 동일하게 모든 속성에 대한 게터와 양식의 모든 정보를 인쇄하는 삽입("") 연산자의 오버로드를 작성합니다.

또한 관료를 매개변수로 받는 `beSigned()` 멤버 함수를 양식에 추가합니다. 이 함수는 관료의 등급이 충분히 높으면(요구 등급보다 높거나 같으면) 양식 상태를 서명됨으로 변경합니다. 1등급이 2등급보다 높다는 것을 기억하세요. 성적이 너무 낮으면 `Form::GradeTooLowException`을 던집니다.

마지막으로 `Bureaucrat`에 `signForm()` 멤버 함수를 추가합니다. 양식에 서명이 완료되면 다음과 같은 내용을 출력합니다:

<관료>가 <양식>에 서명했습니다.


그렇지 않으면 다음과 같은 내용이 인쇄됩니다:

<관료>는 <이유> 때문에 <양식>에 서명할 수 없습니다.

몇 가지 테스트를 구현하고 제출하여 모든 것이 예상대로 작동하는지 확인합니다.

5장

연습 02: 아니요, 28C가 아닌 28B 양식이 필요합니다...

	운동 : 02
아니요, 28C가 아닌 28B 양식이 필요합니다...	
제출 디렉토리 : <i>ex02/</i>	
제출할 파일: Makefile, main.cpp, Bureaucrat.[{h, hpp},cpp], Bureaucrat.cpp + AForm.[{h, hpp},cpp], ShrubberyCreationForm.[{h, hpp},cpp], + 로봇수술 요청 양식.[{h, hpp},cpp], 대통령 사면 양식.[{h, hpp},cpp]	
금지된 기능 : 없음	

이제 기본 양식을 만들었으니 실제로 어떤 기능을 하는 양식을 몇 개 더 만들 차례입니다.

모든 경우에 기본 클래스 Form은 추상 클래스여야 하며, 따라서 이름을 AForm으로 변경해야 합니다. 양식의 속성은 비공개로 유지되어야 하며 기본 클래스에 있어야 한다는 점을 명심하세요.

다음과 같은 구체적인 클래스를 추가합니다:

- **관목 생성 양식:** 필수 등급: 서명 145, 실행 137
작업 디렉터리에 <target>_shrubbery 파일을 생성하고 그 안에 ASCII 트리를 작성합니다.
- **로봇수술 요청 양식:** 필수 성적: 서명 72, 실행 45

드릴링 소음이 발생합니다. 그런 다음 <대상>이 50%의 확률로 로봇 수술에 성공했음을 알립니다. 그렇지 않으면 로봇 수술이 실패했음을 알립니다.

- **대통령 사면 양식:** 필수 등급: 서명 25, 집행 5 <대상>이 자포드 비블록스에 의해 사면되었음을 알립니다.

이들 모두는 생성자에서 양식의 대상인 매개변수를 하나만 받습니다. 예를 들어, 집에 관목을 심으려는 경우 "home"을 사용합니다.

이제 기본 폼에 `execute(Bureaucrat const & executor) const` 멤버 함수를 추가하고 구체적인 클래스에서 폼의 액션을 실행하는 함수를 구현합니다. 폼이 서명되었는지, 폼을 실행하려는 관료의 등급이 충분히 높은지 확인해야 합니다. 그렇지 않으면 적절한 예외를 던지세요.

모든 구체적인 클래스에서 요구 사항을 확인할지, 아니면 기본 클래스에서 확인할지(다른 함수를 호출하여 양식을 실행할지) 여부는 사용자가 결정할 수 있습니다. 하지만 한 가지 방법이 다른 방법보다 낫습니다.

마지막으로, `executeForm(Form const & form)` 멤버 함수를 `Bureau`- 크랫에 추가합니다. 이 함수는 폼을 실행하려고 시도해야 합니다. 성공하면 다음과 같은 내용을 출력합니다:


<관료>가 <양식>을 실행했습니다.

그렇지 않은 경우 명시적인 오류 메시지를 인쇄합니다.

몇 가지 테스트를 구현하고 제출하여 모든 것이 예상대로 작동하는지 확인합니다.

제6장

운동 03: 적어도 커피 만드는 것 보다는 낫습니다.

	운동 : 03
적어도 커피를 만드는 것보다는 낫습니다.	
제출 디렉토리 : <i>ex03/</i>	
제출할 파일 : 이전 연습의 파일 + Intern.{h, hpp}, Intern.cpp	
금지된 기능 : 없음	

양식을 작성하는 것은 충분히 성가신 일이기 때문에 관료들에게 하루 종일 이 일을 하라고 하는 것은 잔인한 일입니다. 다행히도 인턴이 존재합니다. 이 연습에서는 **인턴** 클래스를 구현해야 합니다. 인턴은 이름도 없고, 학년도 없고, 고유한 특성도 없습니다. 관료들이 신경 쓰는 것은 오직 그들이 일을 잘 해내는 것뿐입니다.

하지만 인턴에게는 한 가지 중요한 기능이 있는데, 바로 `makeForm()` 함수입니다. 이 함수는 두 개의 문자열을 받습니다. 첫 번째는 품의 이름이고 두 번째는 품의 대상입니다. 이 함수는 두 번째 매개변수로 초기화될 대상인 **Form 객체**(이름이 매개변수로 전달된 이름)에 대한 포인터를 반환합니다.

다음과 같은 내용이 인쇄됩니다:

인턴이 <양식>을 만듭니다.

매개변수로 전달된 양식 이름이 존재하지 않으면 명시적인 오류 메시지를 인쇄합니다.

if/elseif/else 포레스트를 사용하는 것과 같이 읽기 어렵고 보기 흉한 솔루션은 피해야 합니다. 이런 종류의 것은 평가 과정에서 허용되지 않습니다. 더 이상 피시네(폴)에 있지 않습니다. 평소처럼 모든 것이 예상대로 작동하는지 테스트해야 합니다.

예를 들어, 아래 코드는 "Ben- der"를 대상으로 하는 **RobotomyRequestForm**을 생성합니다:

```
{  
    인턴 일부 랜덤 인턴; 양식*  
        rrf;  
  
    rrf = someRandomIntern.makeForm("로봇 절개 요청", "Bender");  
}
```


제 7장

제출 및 동료 평가

과제는 평소처럼 Git 리포지토리에 제출하세요. 방어 기간 동안에는 리포지토리 내의 작업만 평가됩니다. 폴더와 파일 이름을 다시 한 번 확인하여 정확한지 확인하세요.



16D85ACC441674FBA2DF65190663F9373230CEAB1E4A0818611C0E39F5B26E4D774F1
74620A16827E1B16612137E59ECD492E468A92DCB17BF16988114B98587594D12810
E67D173222A

