бнидия (https://projects.intra.42.fr/scale_teams/3861241/edit#)

(https://profile.intra.42.fr/)

(https://profile.intra.42.fr/searchs)

ШКАЛА ДЛЯ ПРОЕКТА CPP MODULE 04 (HTTPS://PROJECTS.INTRA.42.FR/PROJECTS/CPP-MODULE-04)

Вы должны оценить 1 студента в этой команде

Git-репозиторий

git@vogsphere.msk.21-school.ru:vogsphere/intra-uuid-049049b8-2662-467

Введение

Пожалуйста, соблюдайте следующие правила:

- Оставайтесь вежливым, учтивым, уважительным и конструктивным на протяжении всего процесса оценки. От этого зависит благополучие общества.

- Определите вместе с оцениваемым студентом или группой, чья работа оценивается, возможные недостатки в их проекте. Найдите время, чтобы обсудить и обсудить проблемы, которые, возможно, были выявлены.

- Вы должны учитывать, что могут быть некоторые различия в том, как ваши коллеги могли понять инструкции проекта и объем его функций. Всегда будьте непредвзяты и оценивайте их как можно честнее. Педагогика полезна только и только в том случае, если экспертная оценка проводится серьезно.

Методические рекомендации

- Оценивайте только ту работу, которая была сдана в репозиторий Git оцениваемого учащегося или группы.

- Дважды проверьте, что репозиторий Git принадлежит учащимся. Убедитесь, что проект соответствует ожиданиям. Также убедитесь, что «git clone» используется в пустой папке.

- Внимательно проверьте, чтобы не использовались вредоносные псевдонимы, чтобы обмануть вас и заставить оценить что-то, что не является содержимым официального репозитория.

- Чтобы избежать каких-либо сюрпризов, если применимо, вместе просмотрите все сценарии, используемые для облегчения оценивания (сценарии для тестирования или автоматизации).

- Если вы не выполнили задание, которое собираетесь оценивать, вы должны прочитать всю тему до начала процесса оценивания.

- Используйте доступные флаги, чтобы сообщить о пустом репозитории, неработающей программе, ошибке Norm, читерстве и так далее.
В этих случаях процесс оценки завершается и окончательная оценка равна 0 или -42 в случае списывания. Однако, за исключением списывания, учащимся настоятельно рекомендуется вместе просмотреть сданную работу, чтобы выявить ошибки, которые не следует повторять в будущем.

- Вам никогда не придется редактировать какой-либо файл, кроме файла конфигурации, если он

существует. Если вы хотите отредактировать файл, найдите время, чтобы объяснить причины оцениваемому учащемуся и убедитесь, что вы обасогласны с этим.

- Также необходимо проверить отсутствие утечек памяти. Любая память, выделенная в куче, должна быть должным образом освобождена до окончания выполнения.
Вам разрешено использовать любой из различных инструментов, доступных на компьютере, таких как утечки, valgrind или e_fence. В случае утечек памяти отметьте соответствующий флаг.

Вложения

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/40217/en.subject.pdf)

Предварительные испытания

Если подозревается мошенничество, оценка здесь останавливается. Используйте флаг «Обман», чтобы сообщить об этом. Принимайте это решение спокойно, взвешенно и, пожалуйста, используйте эту кнопку с осторожностью.

Предпосылки

Код должен компилироваться с помощью C++ и флагов -Wall -Wextra -Werror Не забывайте, что этот проект должен соответствовать стандарту C++98. Таким образом, функции или контейнеры C++ 11 (и более поздних версий) НЕ ожидаются.

Любое из этих значений означает, что вы не должны оценивать рассматриваемое упражнение:
- Функция реализована в заголовочном файле (кроме шаблонных функций).

- Makefile компилируется без необходимых флагов и/или другого компилятора, кроме c++.

Любое из этих действий означает, что вы должны пометить проект как «Запрещенная функция»: - Использование функции «C» (*alloc, *printf, free).

- Использование функции, не разрешенной в инструкции по упражнению.
- Использование «использования пространства имен» или ключевого слова «друг».
- Использование внешней библиотеки или функций версий, отличных от C++98.

да Нет

Ex00: Полиморфизм

Как обычно, должна быть основная функция, которая содержит достаточно тестов, чтобы доказать, что программа работает должным образом.
Если нет, не оценивайте это упражнение. Если какой-либо класс, не связанный с интерфейсом, не соответствует ортодоксальной канонической форме класса, не оценивайте это упражнение.

Первая проверка

Существует класс Animal с одним атрибутом: одной строкой с именем type.
Вы должны иметь возможность создавать и использовать этот класс.

да Нет

Наследование

Это как минимум два класса, которые наследуют от Animal: Cat и Dog.
Вых одные данные конструктора и деструктора должны быть четкими.
Спросите студента о порядках конструктора и деструктора.

да Нет

Простой производный класс

Тип атрибута устанавливается в соответствующее значение при создании для каждого животного. У кошки должно быть «Кошка», у собаки должна быть «Собака».

да Нет

Животное

Использование функции makeSound() всегда вызывает соответствующую функцию makeSound(). makeSound() должен быть виртуальным! Проверьте это в коде. virtual void makeSound() const Возвращаемое значение не важно, но ключевое слово virtual является обязательным.

Должен быть пример с WrongAnimal и WrongCat, в которых не используется ключевое слово virtual (см. тему).
WrongCat должен выводить WrongCat makeSound() только при использовании в качестве неправильного кота.

да Нет

Ex01: Я не хочу поджигать мир

Как обычно, должна быть основная функция, содержащая достаточное количество тестов, чтобы доказать, что программа работает должным образом. Если нет, не оценивайте это упражнение. Если какой-либо класс, не связанный с интерфейсом, не соответствует ортодоксальной канонической форме класса, не оценивайте это упражнение.

Бетонное животное

Появился новый класс Brain.

Кошка и Собака имеют обязательный личный атрибут Мозг.
Атрибут Brain не должен находиться внутри класса Animal.

Класс Brain имеет определенные выходные данные при создании и удалении.

да Нет

Бетонный мозг

Копия Кошки или Собаки должна быть глубокой копией.

Тестируйте что-то вроде: Базовая

собака; { Временная температура

собаки = базовая }

Если копия неглубокая, tmp и basic будут использовать один и тот же Brain, а Brain будет удален с tmp в конце области видимости.
Конструктор копирования также должен выполнять глубокое копирование.
Вот почему чистая реализация в ортодоксальной канонической форме избавит вас от многочасовой боли.

да Нет

Цепь разрушения

Деструкторы в Animal и его производных классах являются виртуальными.

Попросите объяснить, что произойдет без виртуального ключевого слова.

Проверьте это.

да Нет

Присвоение и копирование

Копирование и присваивание поведения Кошки и Собаки соответствуют требованиям
субъекта.
Глубокое копирование означает, что вам нужно создать новый мозг для кошки или собаки.
Убедитесь, что каноническая форма действительно реализована (т.е. нет пустых
операторов присваивания копирования и т.д.). Ничто не должно быть публичным без причины.
Более того, этот код очень простой, поэтому его нужно почистить!

да Нет

Ex02: Абстрактный класс

Как обычно, должна быть основная функция, содержащая достаточное количество тестов, чтобы доказать, что программа работает должным образом. Если нет, не оценивайте это упражнение. Если какой-либо класс, не связанный с интерфейсом, не соответствует ортодоксальной канонической форме класса, не оценивайте это упражнение.

Абстрактный класс

Есть класс Animal точно такой же, как в теме.
Animal::makeSound — это чисто виртуальная функция.
Это должно выглядеть так: virtual void makeSound() const = 0; Часть «= 0»
обязательна.
Вы не должны иметь возможность создавать экземпляр Animal.

Тест на животных; //должен дать вам ошибку компиляции о том, что класс является абстрактным

да Нет

Бетонное животное

Классы Cat и Dog по-прежнему присутствуют и работают точно так же, как в ex02.

да Нет

Ex03: Интерфейс и обзор

Как обычно, должна быть основная функция, содержащая достаточное количество тестов, чтобы доказать, что программа работает должным образом. Если нет, не оценивайте это упражнение. Если какой-либо класс, не связанный с интерфейсом, не соответствует ортодоксальной канонической форме класса, не оценивайте это упражнение.

Интерфейсы

Есть интерфейсы ICharacter и IMateriaSource, точно такие же, как и требуется в теме.

да Нет

Источник Материи

Класс MateriaSource присутствует и реализует IMateriaSource. Функции-члены работают по
назначению.

да Нет

Бетонная материя

Существуют конкретные классы Ice и Cure, которые унаследованы от AMateria. Их
Метод clone() реализован корректно. Их выводы верны.
Класс AMateria по-прежнему абстрактен (clone() — это чистая функция).
присутствует виртуальный ~AMateria().
AMateria содержит защищенный строковый атрибут для хранения типа.

да Нет

Персонаж

Класс Character присутствует и реализует ICharacter. Она имеет
инвентарь не более 4 Материй.
Функции-члены реализуются в соответствии с требованиями субъекта.
Копия и назначение Персонажа реализованы по мере необходимости.
(глубокая копия).

да Нет

Рейтинги

Не забудьте проверить флаг, соответствующий защите

Ok

Выдающийся проект

Пустая работа Нет файла автора W Недопустимая компиляция Стандарты Изменять г Сбой

Утечки л Запрещенная функция

Заключение

Оставьте комментарий к этой оценке

Завершить оценку