



C++ - Модуль 03

Наследование

Резюме: Этот документ содержит упражнения Модуля 03 из модулей C++.

Версия : 6

Содержание

| | | |
|-----|-------------------------------------|---|
| я | Введение | 2 |
| ил | Основные правила | 3 |
| III | Упражнение 00: Ааааа... ОТКРЫТ Ы! | 5 |
| IV | Упражнение 01: Серена, любовь моя ! | 7 |
| V | Упражнение 02: Повторяюся работа | 8 |
| VI | Упражнение 03: Теперь это странно! | 9 |

Глава I

Введение

C++ — это язык программирования общего назначения, созданный Бьерном Страуструпом как расширение языка программирования C или «C с классами» (источник: [Википедия](#)).

Цель этих модулей — познакомить вас с объектно-ориентированным программированием. Это будет отправной точкой вашего путешествия по C++. Многие языки рекомендуются для изучения ООП. Мы решили выбрать C++, так как она является производным от вашего старого знакомого C. Поскольку это сложный язык, и для простоты ваш код будет соответствовать стандарту C++98.

Мы знаем, что современный C++ сильно отличается во многих аспектах. Так что, если вы хотите стать опытным разработчиком C++, вам решать, идти ли дальше после 42 Common Core!

Глава II

Основные правила

Компиляция

- Скомпилируйте свой код с помощью C++ и флагов `-Wall -Wextra -Werror`
- Ваш код все равно должен компилироваться, если вы добавите флаг `-std=c++98`.

Соглашения о форматировании и именованиях

- Каталог и упражнения будут называться следующим образом: `ex00`, `ex01`, ..., `exn`
- Назовите свои файлы, классы, функции, функции-члены и атрибуты, как требуется в рекомендациях.
- Пишите имена классов в формате `UpperCamelCase`. Файлы, содержащие код класса, будут всегда называться в соответствии с именем класса. Например: `ClassName.hpp/ClassName.h`, `ClassName.cpp` или `ClassName.tpp`. Затем, если у вас есть заголовочный файл, содержащий определение класса «BrickWall», обозначающего кирпичную стену, его имя будет `BrickWall.hpp`.
- Если не указано иное, каждое выходное сообщение должно заканчиваться символом новой строки. Символ отображается на стандартный вывод.
- Досвидания, Норминетт! В модулях C++ не применяется стиль кодирования. Вы можете следить за своим любимым. Но имейте в виду, что код, который не могут понять ваши коллеги-оценщики, — это код, который они не могут оценить. Старайтесь писать чистый и читаемый код.

Разрешено/Запрещено

Вы больше не кодируете на C. Время C++! Следовательно:

- Вам разрешено использовать почти все из стандартной библиотеки. Таким образом, вместо того, чтобы придерживаться того, что вы уже знаете, было бы разумно использовать как можно больше C++-версий функций C, к которым вы привыкли.
- Однако вы не можете использовать никакую другую внешнюю библиотеку. Это означает, что C++11 (и производные формы) и библиотеки Boost запрещены. Также запрещены следующие функции: `*printf()`, `*alloc()` и `free()`. Если вы их используете, ваша оценка будет 0 и все.

- Обратите внимание, что если явно не указано иное, используемое пространство имен `<ns_name>` и ключевые слова друзей запрещены. В противном случае ваша оценка будет -42.
- Вам разрешено использовать STL только в Модуле 08. Это означает: никаких контейнеров (вектор/список/карта/и т. д.) и никаких алгоритмов (все, что требует включения заголовка `<algorithm>`) до тех пор. В противном случае ваша оценка будет -42.

Несколько требований к дизайну

- Утечка памяти происходит и в C++. Когда вы выделяете память (используя новый ключевое слово), вы должны избегать утечек памяти.
- От Модуля 02 до Модуля 08 ваши занятия должны быть оформлены в правовом стиле. Каноническая форма, за исключением случаев, когда прямо указано иное.
- Любая реализация функции, помещенная в заголовочный файл (кроме шаблонов функций), означает 0 для упражнения.
- Вы должны иметь возможность использовать каждый из ваших заголовков независимо от других. Таким образом, они должны включать все необходимые им зависимости. Однако вы должны избежать проблемы двойного включения, добавив защиту включения. В противном случае ваша оценка будет 0.

Прочтите меня

- При необходимости вы можете добавить несколько дополнительных файлов (например, для разделения кода). Поскольку эти назначения не проверяются программой, не стесняйтесь делать это, пока вы не дадите окончательные файлы.
- Иногда рекомендации к упражнению кажутся короткими, но примеры могут показать требования, которые явно не прописаны в инструкциях.
- Полностью прочитайте каждый модуль перед началом! Действительно, сделайте это.
- Клянусь Одном, клянусь Твором! Используйте свой мозг !!!




Вам придется реализовать много классов. Это может показаться утомительным, если только вы не умеете писать сценарии в своем любимом текстовом редакторе.



Вам предоставляется определенная свобода для выполнения упражнений. Однако соблюдайте обязательные правила и не ленитесь. Вы будете упускать много полезной информации! Не стесняйтесь читать о теоретических концепциях.

Глава III

Упражнение 00: Аааааа... ОТ КРЫГ Ы!

| | |
|---|-----------------|
|  | Упражнение : 00 |
| Ааааа... ОТ КРЫГ Ы! | |
| Каталог с дачи: ex00/ Файлы | |
| для с дачи: Makefile, main.cpp, ClapTrap.{h, hpp}, ClapTrap.cpp Запрещенные функц ии: нет | |
| | |

Во-первых , вы должны реализовать клас с ! Как ориг инально!

Он будет называться ClapTrap и будет иметь следующие частные атрибуты, инициализированные значениями, указанными в скобках :

- Имя , которое передается в качестве параметра конструктору .
- Очки жизни (10) представля ют собой здоровье ClapTrap.
- Очки энергии (10)
- Урон от атаки (0)

Добавьте следующие общедоступные функции-члены, чтобы ClapTrap выгля дел более реалистично:

- недействительная атака(const std::string& target);
- void takeDamage(беззнаковая целая сумма);
- void beRepaired(беззнаковая целая сумма);

Когда ClapTrap атакует, цель теряет хиты на <урон от атаки>.

Когда ClapTrap восстанавливает себя , он получает обратно <количество> очков жизни. Атака и ремонт стоят 1 очко энергии каждое. Конечно, ClapTrap ничего не может сделать, если у него не осталось ни очков жизни, ни очков энергии.

Во всех этих функциях-членах вы должны вывести сообщение, описывающее, что происходит. Например, функция `Attack()` может отображать что-то вроде (разумеется, без угловых скобок):


`ClapTrap <name> атакует <цель>, нанося <damage> единиц урона!`

Конструкторы и деструкторы также должны отображать сообщение, чтобы ваши коллеги-оценщики могли легко увидеть, что они были вызваны.

Внедрите и дайте собственные тесты, чтобы убедиться, что ваш код работает должным образом.

Глава IV

Упражнение 01: Серена, любовь моя !

| | |
|--|-----------------|
|  | Упражнение : 01 |
| Серена, любовь моя ! | |
| Каталог : с дачи: ex01/ Файлы | |
| для : с дачи: файлы из предыдущего упражнения + ScavTrap.{h, hpp}, ScavTrap.cpp Запрещенные функции: нет | |
| | |

Поскольку у вас никогда не будет достаточно ClapTraps, теперь вы создадите производного робота. Он будет называться ScavTrap и унаследует конструкторы и деструктор от ClapTrap. Однако его конструкторы, деструктор и Attack() будут печатать разные сообщения. Ведь ClapTraps осознают свою индивидуальность.

Обратите внимание, что в ваших тестах должна быть показана правильная цепочка построения /разрушения. Когда создается ScavTrap, программа начинается с создания ClapTrap. Разрушение происходит в обратном порядке. Почему?

ScavTrap будет использовать атрибуты ClapTrap (после этого обновите ClapTrap) и должны инициализировать их, чтобы:

- Имя, которое передается в качестве параметра конструктору.
- Очки жизни (100) представляющие собой здоровье ClapTrap.
- Очки энергии (50)
- Урон от атаки (20)

У ScavTrap также будет своя собственная функция:


```
аннулировать GuardGate();
```

Эта функция-член отобразит сообщение, информирующее о том, что ScavTrap теперь находится в режиме привратника.

Не забудьте добавить больше тестов в вашу программу.

Глава V

Упражнение 02: Повторяющаяся работа

| | |
|---|---|
|  | Упражнение : 02 |
| Повторяющаяся работа | |
| Каталог с дачи: ex02/ Файлы | для с дачи: файлы из предыдущих упражнений + FragTrap.{h, cpp}, FragTrap.cpp Запрещенные функции: нет |
| | |

Создание ClapTraps, вероятно, начинает действовать вам на нервы.

Теперь реализуйте класс FragTrap, наследуемый от ClapTrap. Он очень похож на ScavTrap. Однако его сообщения о построении и уничтожении должны быть разными. Правильная цепочка построения /разрушения должна быть показана в ваших тестах. При создании FragTrap программа начинается с создания ClapTrap. Разрушение происходит в обратном порядке.

То же самое для атрибутов, но не с теми значениями:

- Имя, которое передается в качестве параметра конструктору.
- Очки жизни (100) представляющие собой здоровье ClapTrap.
- Очки энергии (100)
- Урон от атаки (30)

FragTrap также имеет особую особенность:


```
недействительным highFivesGuys (недействительным);
```

Эта функция-член отображает положительный запрос «дай пять» в стандартном выводе.

Опять же, добавьте больше тестов в свою программу.

Глава VI

Упражнение 03: Теперь это странно!

| | |
|--|-----------------|
|  | Упражнение : 03 |
| Теперь это странно! | |
| Каталог с данными: ex03/ Файлы для | |
| с данными: файлы из предыдущих упражнений + DiamondTrap.{h, cpp}, DiamondTrap.cpp Запрещенные функции: нет | |
| | |

В этом упражнении вы создадите монстра ClapTrap, наполовину FragTrap, наполовину ScavTrap. Он будет называться DiamondTrap и будет унаследован как от FragTrap, так и от ScavTrap. Это так рисковано!

Класс DiamondTrap будет иметь закрытый атрибут name. Дайте этому атрибуту точно такое же имя переменной (не говоря уже об имени робота), что и в базовом классе ClapTrap.

Чтобы было понятнее, вот два примера.

Если переменная ClapTrap имеет имя, присвойте имя переменной DiamondTrap.

Если переменная ClapTrap имеет значение _name, присвойте имя _name переменной DiamondTrap.

Его атрибуты и функции-члены будут выбраны из любого из его родительских классов:

- Имя, которое передается в качестве параметра конструктору.
- ClapTrap::name (параметр конструктора + суффикс "_clap_name")
- Очки жизни (FragTrap)
- Очки энергии (ScavTrap)
- Урон от атаки (FragTrap)
- атака() (Scavtrap)

Помимо специальных функций обоих родительских классов, DiamondTrap будет иметь свои собственные возможности:

аннулировать whoAmI();

Эта функция-член будет отображать как свое имя, так и имя ClapTrap.

Конечно, объект ClapTrap для DiamondTrap будет создан один раз и только один раз. Да, есть хитрость.

Опять же, добавьте больше тестов в свою программу.



Знакомы ли вы с флагами компилятора -Wshadow и -Wno-shadow?



Вы можете пройти этот модуль, не выполняя упражнение 03.