



C++ - Модуль 06

C++ приводит

Резюме: Этот документ содержит упражнения Модуля 06 из модулей C++.

Версия : 5

Содержание

я	Введение	2
ил	Основные правила	3
III	Дополнительное правило	5
IV	Упражнение 00. Преобразование скалярных типов	6
V	Упражнение 01: Сериализация	8
VI	Упражнение 02. Определите реальный тип	9

Глава I

Введение

C++ — это язык программирования общего назначения, созданный Бьерном Страуструпом как расширение языка программирования C или «C with Classes» (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 прочтите 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


Дополнительное правило

Следующее правило применяется ко всему модулю и не является обязательным.

Для каждого упражнения преобразование типов должно быть решено с использованием одного конкретного типа приведения. Ваш выбор будет проверен во время защиты.

Глава IV

Упражнение 00: Преобразование скалярных типов

	Упражнение 00
Преобразование скалярных типов	
Каталог с дачи: ex00/	
Файлы для с дачи: Makefile, *.cpp, *.{h, hpp}	
Разрешенные функции: любая функция для преобразования строки в int, float или double. Это поможет, но не сделает всю работу.	

Напишите программу, которая принимает в качестве параметра строковое представление литерала C++ в его наиболее распространенной форме. Этот литерал должен принадлежать к одному из следующих скалярных типов: char, int, float или double. За исключением параметров char, будет использоваться только десятичная запись.

Примеры символьных литералов: 'c', 'a', ...

Для простоты обратите внимание, что не отображаемые символы не должны использоваться в качестве входных данных. Если преобразование в char не отображается, выводит информационное сообщение.

Примеры литералов int: 0, -42, 42...

Примеры литералов с плавающей запятой: 0.0f, -4.2f, 4.2f...

Вы также должны обрабатывать типичные литералы (вы знаете, для науки): -inff, +inff и nanf.

Примеры двойных литералов: 0.0, -4.2, 4.2...

Вы также должны обрабатывать типичные литералы (вы знаете, для развлечения): -inf, +inf и nan.


Вы должны с начала определить тип литерала, переданного в качестве параметра, преобразовать его из строки в фактический тип, а затем явно преобразовать в три других типа данных. Наконец, отобразите результаты, как показано ниже.

Если преобразование не имеет смысла или приводит к переполнению отобразите сообщение, информирующее пользователя о том, что преобразование типа невозможно. Включите любой заголовок, который вам нужен для обработки числовых ограничений и специальных значений.

```
./convert 0 char:  
Не отображается int: 0 float: 0.0f  
  
double: 0.0 ./  
convert nan char:  
невозможно int: невозможно  
float: nanf  
  
двойной: в  
./конвертировать 42.0f  
символ: '*'  
интервал: 42  
поплавок: 42.0f  
двойной: 42.0
```


Глава V

Упражнение 01. Сериализация

	Упражнение : 01
Сериализация	
Каталог с дачи: ex01/ Файлы	
для с дачи: Makefile, *.cpp, *.h, hpp}	
Запрещенные функц ии: нет	

Реализовать следующие функц ии:

`uintptr_t` сериализовать (данные `* ptr`);

Он берет указатель и преобразует его в беззнаковый целочисленный тип `uintptr_t`.

Данные `* raw` десериализовать (`uintptr_t raw`);

Он принимает беззнаковый целочисленный параметр и преобразует его в указатель на данные.

Напишите программу для проверки правильности работы ваших функц ий.


Вы должны создать непустую(это означает, что она имеет элементы данных) структуру данных.

Используйте `serialize()` для адреса объекта `Data` и передайте его возвращаемое значение в `deserialize()`.
Затем убедитесь, что возвращаемое значение `deserialize()` совпадает с исходным указателем.

Не забудьте дать файлы вашей структуры данных.

Глава VI

Упражнение 02. Определите настоящий тип

	Упражнение : 02
Определить настоящий тип	
Каталог с дачи: ex02/ Файлы	
для с дачи: Makefile, *.cpp, *.{h, hpp}	
Запрещенные функции: std::typeinfo	

Реализуйте базовый класс, который имеет только общедоступный виртуальный деструктор. Создайте три пустых класса A, B и C, которые публично наследуются от Base.



Эти четыре класса обязательно должны быть разработаны в правильном канонической форме.

Реализовать следующие функции:

База * генерировать (недействительно); Он случайным образом создает экземпляры A, B или C и возвращает экземпляр в качестве базового указателя. Не стесняйтесь использовать все, что вам нравится для реализации с случайного выбора.

недействительная идентификация (база * p); Он печатает фактический тип объекта, на который указывает p: «A», «B» или «C».

недействительная идентификация (база * p); Он печатает фактический тип объекта, на который указывает p: «A», «B» или «C». Использование указателя внутри этой функции запрещено.

Включение заголовка typeid запрещено.

Напишите программу для проверки того, что все работает как положено.