



C++ - Модуль 02

Ad-hoc полиморфизм, перегрузка операторов и ортодоксальная каноническая форма класса

Резюме:

Этот документ содержит упражнения по созданию модуля из модулей 02C++.

Версия: 7

Содержание

I	Введение	2
II	Общие правила	3
III	Новые правила	5
IV	Упражнение 00: Мой первый урок в православной канонической форме	6
V	Упражнение 01: На пути к более полезному классу чисел с фиксированной точкой	8
VI	Упражнение 02: Теперь мы говорим	10
VII	Упражнение 03: BSP	12

Глава I

Введение

С++ - это язык программирования общего назначения, созданный Бьярном Струstrupом как продолжение языка программирования С, или "С с классами" (источник: [Википедия](#)).

Цель этих модулей - познакомить вас с **объектно-ориентированным программированием**. Это будет отправной точкой вашего путешествия по С++. Многие языки рекомендуются для изучения ООП. Мы решили выбрать С++, поскольку он является производным от вашего старого друга С. Поскольку это сложный язык, и для того, чтобы все было просто, ваш код будет соответствовать стандарту С++98.

Мы понимаем, что современный С++ во многих аспектах сильно отличается. Поэтому, если вы хотите стать квалифицированным разработчиком С++, вам предстоит пройти дальше 42 Common Core!

Глава II

Общие правила

Компиляция

- Скомпилируйте ваш код с помощью c++ и флагов -Wall -Wextra -Werror
- Ваш код будет компилироваться, если вы добавите флаг -std=c++98

Форматирование и соглашения об именовании

- Каталоги упражнений будут называться так: ex00, ex01, ..., exn
- Назовите свои файлы, классы, функции, функции-члены и атрибуты в соответствии с требованиями руководства.
- Записывайте имена классов в формате **UpperCamelCase**. Файлы, содержащие код класса, всегда будут именоваться в соответствии с именем класса. Например: ClassName.hpp/ClassName.h, ClassName.cpp или ClassName.hpp. Тогда, 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 завершаться символом новой строки и выводиться на стандартный вывод.
- *До свидания, Норминет!* В модулях C++ нет принудительного стиля кодирования. Вы можете следовать своему любимому стилю. Но имейте в виду, что код, который ваши коллеги-оценщики не могут понять, они не могут оценить. Делайте все возможное, чтобы писать чистый и читабельный код.

Разрешено/Запрещено

Вы больше не кодируете на C. Пора переходить на C++! Поэтому:

- 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++-шных версий функций языка C, к которым вы привыкли.
- Однако вы не можете использовать никакие другие внешние библиотеки.

Это означает, что библиотеки C++11 (и производные формы) и 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-друзья запрещены. В противном случае ваша оценка будет равна -42.
- **Вам разрешено использовать STL только 08 в модуле.** Это означает: никаких **контейнеров** (вектор/лист/мап/ и так далее) и никаких **алгоритмов** (все, что требует включения заголовка `<algorithm>`) до этого момента. В противном случае ваша оценка будет -42.

Несколько требований к дизайну

- Утечка памяти происходит и в C++. Когда вы выделяете память (с помощью функции `new` ключевое слово), вы должны избегать **утечек памяти**.
- С модуля 02 по модуль 08 ваши занятия должны быть построены в **православной канонической форме, за исключением случаев, когда прямо указано иное**.
- Любая реализация функции, помещенная в заголовочный файл (кроме шаблонов функций), означает 0упражнение.
- Вы должны иметь возможность использовать каждый из ваших заголовков независимо от других. Таким образом, они должны включать все необходимые зависимости. Однако вы должны избегать проблемы двойного включения, добавляя **защитные элементы include**. В противном случае ваша оценка будет следующей 0.

Читать

- Вы можете добавить несколько дополнительных файлов, если это необходимо (например, для разделения вашего кода). Поскольку эти задания не проверяются программой, не стесняйтесь делать это, если вы сдаете обязательные файлы.
- Иногда указания к упражнению выглядят кратко, но на примерах можно увидеть требования, которые не прописаны в инструкциях в явном виде.
- Перед началом работы полностью прочитайте каждый модуль! Действительно, сделайте это.
- Одином, Тором! Используйте свой мозг!!!



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



Вам предоставляется определенная свобода в выполнении упражнений. Однако соблюдайте обязательные правила и не ленитесь. Иначе вы пропустите много полезной информации! Не стесняйтесь читать о теоретических концепциях.

Глава III

Новые правила


С этого момента все ваши классы должны быть спроектированы в **ортодоксальной канонической форме**, если явно не указано иное. Тогда они будут реализовывать четыре необходимые функции-члена, приведенные ниже:

- Конструктор по умолчанию
- Конструктор копирования
- Оператор присвоения копий
- Деструктор

Разделите код вашего класса на два файла. Заголовочный файл (.hpp/.h) содержит определение класса, а исходный файл (.cpp) - реализацию.

Глава IV

Упражнение 00: Мой первый урок в православной канонической форме

	Упражнение 00
Мой первый урок в православной канонической форме	
Входящий каталог : <code>ex/00</code>	
Файлы для сдачи : <code>Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp</code>	
Запрещенные функции : Нет	

Вы думаете, что знаете целые числа и числа с плавающей точкой. Как мило.

Пожалуйста, прочитайте статью на этой странице [3 \(123\)](#), чтобы понять, что это не так. Продолжайте, читайте.

До сегодняшнего дня каждое число, которое вы использовали в своем коде, в основном было либо целым числом, либо числом с плавающей точкой, либо любым из их вариантов (`short`, `char`, `long`, `double` и так далее). Прочитав статью выше, можно с уверенностью предположить, что целые числа и числа с плавающей запятой имеют противоположные характеристики.

Но сегодня все изменится. Вы откроете для себя новый удивительный тип чисел: **числа с фиксированной точкой**! Вечно отсутствующие в скалярных типах большинства языков, числа с фиксированной точкой предлагают ценный баланс между производительностью, точностью, диапазоном и точностью. Это объясняет, почему числа с фиксированной точкой особенно применимы в компьютерной графике, обработке звука или научном программировании.

Поскольку в C++ нет чисел с фиксированной точкой, вы будете их складывать. [Эта статья](#) из Беркли - хорошее начало. Если вы понятия не имеете, что такое университет Беркли, прочитайте [этот раздел](#) его страницы в Википедии.

Создайте класс в ортодоксальной канонической форме, который представляет число с фиксированной точкой:

- Рядовые члены:
 - **Целое число** для хранения значения числа с фиксированной точкой.
 - **Статическая константа целого числа** для хранения количества дробных битов. Ее значением всегда будет целочисленный литерал 8.
- Общественные члены:
 - Конструктор по умолчанию, который инициализирует значение числа с фиксированной точкой в 0.
 - Конструктор копий.
 - Перегрузка оператора присвоения копий.
 - Деструктор.
 - Функция-член `int getRawBits(void) const;` который возвращает необработанное значение величины с фиксированной точкой.
 - Функция-член `void setRawBits(int const raw);` который устанавливает необработанное значение числа с фиксированной точкой.

Выполняем этот код:

```
#include <iostream>

int main( void ) {

    Исправлено
    a;
    Исправлено
    b( a );
    Исправлено
    c;

    c = b;

    std::cout << a. getRawBits() << std::endl;
    std::cout << b. getRawBits() << std::endl;
} std::cout << c. getRawBits() << std::endl;


    возвращение0
```

Должно получиться что-то похожее на:

```
$> ./a.out
Конструктор по
умолчанию вызывается
Конструктор
копирования вызывается
Вызывается оператор присвоения копий // <-- Эта строка может отсутствовать в зависимости от
вашей реализации вызывается функция-член getRawBits
Конструктор по умолчанию вызывается
Оператор присвоения копий
вызывается функцией-членом
getRawBits вызывается функцией-
членом getRawBits вызывается 0
вызывается функция-член
getRawBits 0
вызывается функция-член
getRawBits 0
Деструктор
называется
Деструктор
называется
Деструктор
называется
$>
```

Глава V

Упражнение 01: На пути к более полезному классу чисел с фиксированной точкой

	Упражнение 01
На пути к более полезному классу чисел с фиксированной точкой	
Входящий каталог : <code>ex/01</code>	
Файлы для сдачи : <code>Makefile</code> , <code>main.cpp</code> , <code>Fixed.{h, hpp}</code> , <code>Fixed.cpp</code>	
Разрешенные функции : <code>roundf</code> (из <code><cmath></code>)	

Предыдущее упражнение было хорошим началом, но наш класс довольно бесполезен. Он может представлять только значение 0.0.

Добавьте в свой класс следующие публичные конструкторы и публичные

- функции-члены: Конструктор, принимающий в качестве параметра целое число.
Он преобразует его в соответствующее значение с фиксированной точкой. Значение дробных битов инициализируется, как 8в упражнении 00.
- Конструктор, принимающий в качестве параметра постоянное число с плавающей точкой.
Он преобразует его в соответствующее значение с фиксированной точкой. Значение дробных битов инициализируется, как 8в упражнении 00.
- Функция-член `float toFloat(void) const;`
который преобразует значение с фиксированной точкой в значение с плавающей точкой.
- Функция-член `int toInt(void) const;`
который преобразует значение с фиксированной точкой в целочисленное значение.

И добавьте следующую функцию в файлы класса **Fixed**:

- Перегрузка оператора `insertion (""),` который вставляет представление числа с

плавающей точкой в объект выходного потока, переданный в качестве параметра.

Выполнение

этого кода:

```
#include <iostream>

int main( void ) {

    Испр      a;
    Испр      b( 10 ); Испр      c( 42.42f );
    Испр      d( b );

    a = Fixed( 1234.4321f );

    std::cout << "a is " << a << std::endl;
    std::cout << "b is " << b << std::endl;
    std::cout << "c is " << c << std::endl;
    std::cout << "d is " << d << std::endl;

    std::cout << "a is " << a.toInt() << " as integer" << std::endl;
    std::cout << "b is " << b.toInt() << " as integer" << std::endl;
    std::cout << "c is " << c.toInt() << " как целое число" <<
    std::endl; std::cout << "d есть " << d.toInt() << " как целое
}      число" << std::endl;


    возвращение 0
}
```

Должно получиться что-то похожее на:

```
$> ./a.out
Конструктор по
умолчанию называется
Int конструктор
называется Float
конструктор называется
Сору конструктор
называется
Оператор присвоения копий
называется конструктор Float
называется
Оператор присвоения копий
называется Деструктор
называется
a есть 1234.43
b - это 10
c - это 42.4219
d - это 10
a является
1234целым
числом b является
10целым числом
c - 42.4219
число d - как 10
целое число
Деструктор
называется
Деструктор
называется
Деструктор
называется
Деструктор
называется
Деструктор
называется
$>
```


Глава VI

Упражнение 02: Теперь мы говорим

	Упражнени е 02
Сейчас мы говорим	
Входящий каталог : ex/02	
Файлы для сдачи : Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp	
Разрешенные функции : roundf (из <cmath>)	

Добавьте в свой класс функции-члены public, чтобы перегрузить следующие операторы:

- Операторы сравнения6: >, <, >=, <=, == и !=.
- Арифметические4 операторы: +, -, * и /.
- 4 оператора инкремента/декремента (пре-инкремент и пост-инкремент, пре-декремент и пост-декремент), которые будут увеличивать или уменьшать значение с фиксированной точкой от наименьшего представимого значения E , например, $+1E > 1$.

Добавьте эти четыре публичные перегруженные функции-члена в свой класс:

- Статическая функция-член min, которая принимает в качестве параметров две ссылки на числа с фиксированной точкой и возвращает ссылку на наименьшее из них.
- Статическая функция-член min, которая принимает в качестве параметров две ссылки на **константу** чисел с фиксированной точкой и возвращает ссылку на наименьшее из них.
- Статическая функция-член max, которая принимает в качестве параметров две ссылки на числа с фиксированной точкой и возвращает ссылку на наибольшее из них.
- Статическая функция-член max, которая принимает в качестве параметров две ссылки на **константу**

чисел с фиксированной точкой и возвращает ссылку на наибольшее из них.

Вы сами должны проверить каждую функцию вашего класса. Однако, выполнив приведенный ниже код:

```
#include <iostream>

int main( void ) {

    Исправлен    a;
    о            b( Fixed( 5.05f ) * Fixed( )2 );
    Исправлен
    std::cout << a << std::endl;
    std::cout << ++a << std::endl;
    std::cout << a << std::endl;
    std::cout << a++ << std::endl;
    std::cout << a << std::endl;

    std::cout << b << std::endl;

    std::cout << Fixed::max( a, b ) << std::endl;

    возвращение0
}

```

Должно получиться что-то вроде (для большей читабельности в приведенном ниже примере убраны смыслы конструктора/деструктора):


```
$> ./a.out
0
0.00390625
0.00390625
0.00390625
0.0078125
10.1016
10.1016
$>

```


Глава VII

Упражнение 03:

BSP

	Упражнение 03
BSP	
Входящий каталог : <code>ex/03</code>	
Файлы для сдачи : <code>Makefile</code> , <code>main.cpp</code> , <code>Fixed.{h, hpp}</code> , <code>Fixed.cpp</code> , <code>Point.{h, hpp}</code> , <code>Point.cpp</code> , <code>bsp.cpp</code>	
Разрешенные функции : <code>roundf</code> (из <code><cmath></code>)	

Теперь, когда у вас есть функциональный класс **Fixed**, было бы неплохо его использовать.

Реализуйте функцию, которая показывает, находится ли точка внутри треугольника или нет.

Очень полезно, не так ли?



BSP расшифровывается как Binary space partitioning.

Пожалуйста. :)



Вы можете пройти этот модуль без выполнения упражнений 03.

Начнем с создания класса **Point** в ортодоксальной канонической форме, который представляет двумерную точку:

- Рядовые члены:
 - Фиксированный const-атрибут x.
 - Фиксированный const-атрибут y.
 - Что-нибудь еще полезное.
- Общественные члены:
 - Конструктор по умолчанию, который инициализирует x и y значениями 0.
 - Конструктор, принимающий в качестве параметров два постоянных числа с плавающей точкой. Он инициализирует x и y этими параметрами.
 - Конструктор копий.
 - Перегрузка оператора присвоения копий.
 - Деструктор.
 - Что-нибудь еще полезное.

В заключение реализуйте следующую функцию в соответствующем файле:

```
bool bsp( Point const a, Point const b, Point const c, Point const point);
```

- a, b, c: Вершины нашего любимого треугольника.
- точка: Точка для проверки.
- Возвращает: True, если точка находится внутри треугольника. Ложь в противном случае. Таким образом, если точка является вершиной или ребром, возвращается False.

Выполните и сдайте собственные тесты, чтобы убедиться, что ваш класс ведет себя так, как ожидается.

