



C++ - Модуль 02

Специальный полиморфизм, перегрузка операторов и
ортодоксальная каноническая форма класса

Резюме:

Этот документ содержит упражнения Модуля 02 из модулей C++.

Версия: 7

Содержание

я	Введение	2
ил	Основные 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 поговорим	10
VII	Упражнение 03: BSP	12

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 он 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 именовании

- Каталоги упражнений будут называться следующим образом: `ex00`, `ex01`, ..., `exp`
- Назовите свои файлы, классы, функции, функции-члены и атрибуты, как требуется в рекомендации.
- 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 и отображается на стандартный вывод.
- До свидания, 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 православном 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.

Прочти 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 сдаете обязательные 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, сделай это.
- Клянусь Одним, клянусь Тором! Используй 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. Ты бы упускаю много полезной информации! Не стесняйтесь читать о теоретические концепции.

Глава III

Новые правила


Отныне все ваши занятия должны быть оформлены в православной канонической форме, если прямо не указано иное. Затем они будут реализовывать четыре обязательные функции-члены ниже:

- Конструктор по умолчанию
- Конструктор копирования
- Копировать оператор присваивания
- Деструктор

Разделите код класса на два файла. Заголовочный файл (.hpp/.h) содержит класс определение, тогда как исходный файл (.cpp) содержит реализацию.

Глава IV

Упражнение 00: Мой первый урок в Православная каноническая форма

	Упражнение : 00
Мой первый класс в православной канонической форме	
Каталог сдачи: ex00/ Файлы для	
сдачи: Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp Запрещенные функции: нет	

Вы думаете, что знаете целые числа и числа с плавающей запятой. Какой милый.

Пожалуйста, прочтите эту 3-страничную статью ([1](#), [2](#), [3](#)) обнаружить, что это не так. Давай, читай.

До сегодняшнего дня каждое число, которое вы использовали в своем коде, было либо целым числом, либо числом с плавающей запятой, либо любым из их вариантов (short, char, long, double и т. д.). Прочитав приведенную выше статью, 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! Навсегда отсутствующие в скалярных типах большинства языков, числа с фиксированной запятой предлагают ценный баланс между производительностью, точностью, диапазоном и точностью. Это объясняет, почему числа с фиксированной точкой особенно применимы к компьютерной графике, обработке звука или научному программированию, и это лишь некоторые из них.

Поскольку в C++ отсутствуют числа с фиксированной точкой, вы собираетесь их добавить. [эта статья](#) из Беркли — хорошее начало. Если вы не знаете, что такое Университет Беркли, прочтите [этот раздел](#) на его странице в Википедии.

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

- Частные члены:

- Целое число для хранения значения числа с фиксированной точкой.
- Статическое постоянное целое число для хранения количества дробных битов. Его значением всегда будет целочисленный литерал 8.

- Общественные члены:

- Конструктор по умолчанию, который инициализирует числовое значение с фиксированной точкой равным 0.
- Конструктор копирования.
- Перегрузка оператора присваивания копии.
- Деструктор.
- Функция-член `int getRawBits(void) const`; который возвращает необработанное значение значения с фиксированной точкой.
- Функция-член `void setRawBits(int const raw)`; который устанавливает необработанное значение числа с фиксированной точкой.

Запуск этого кода:

```
#include <iomanip>

ИНТ          основной ( недействительный ) {

    исправлено;
    Фиксированный b(a);
    Фиксированный c;

    c = b;

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

    вернуть 0;
}
```

Должно вывести что-то похожее на:


```
$> ./a.out
Вызывается конструктор по умолчанию
Вызывается конструктор копирования
Оператор присваивания копирования, называемый // <-- Эта строка может отсутствовать в зависимости от вашей реализации
функции-члена getRawBits, называемой
Вызывается конструктор по умолчанию
Оператор присваивания копирования с
именем функция-член getRawBits с именем
функция-член getRawBits с именем 0 функция-
член getRawBits с именем 0

Функция-член getRawBits называется 0

Вызван деструктор
Вызван деструктор
Вызван деструктор
$>
```


Глава V

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

	Упражнение 01
<p>На пути к более полезному классу чисел с фиксированной точкой</p> <p>Каталог сдачи: ex01/ Файлы для сдачи: Makefile, main.cpp, Fixed.{h, cpp}, Fixed.cpp Разрешенные функции: roundf (из <cmath>)</p>	

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

Добавьте в класс следующие общедоступные конструкторы и общедоступные функции-члены:

- Конструктор, принимающий в качестве параметра постоянное целое число .
Он преобразует его в соответствующее значение фиксированной точки. Дробное значение битов инициализируется равным 8, как в упражнении 00.
- Конструктор, принимающий в качестве параметра постоянное число с плавающей запятой .
Он преобразует его в соответствующее значение фиксированной точки. Дробное значение битов инициализируется равным 8, как в упражнении 00.
- Функция-член `float toFloat(void) const;`
который преобразует значение с фиксированной запятой в значение с плавающей запятой.
- 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 вставки (`<<`), который вставляет представление числа с фиксированной точкой в формате с плавающей запятой в объект выходного потока, переданный в качестве параметра.

Запуск этого кода:

```
#include <iostream>

ИНТ    основной ( недействительный ) {

    фиксированный    а;
    Фиксированная константа b( 10 );
    фиксированная константа c( 42.42f );
    Фиксированная константа d(b);

    а = фиксированный ( 1234.4321f );

    std::cout << "а — это"      " << а << std::endl;
    std::cout << "b — это"      " << b << std::endl;
    std::cout << "c — это"      " << c << std::endl;
    std::cout << "d — это"      " << d << std::endl;

    std::cout << "а есть"      " << a.toInt() << " как целое число" << std::endl;
    std::cout << "b — это"      " << b.toInt() << " как целое число" << std::endl;
    std::cout << "c — это"      " << c.toInt() << " как целое число" << std::endl;
    std::cout << "d — это"      " << d.toInt() << " как целое число" << std::endl;


    вернуть 0;
}
```

Должно вывести что-то похожее на:

```
$> ./a.out
Вызывается конструктор по умолчанию
Конструктор Int вызывается
Вызывается конструктор с плавающей запятой
Вызывается конструктор копирования
Оператор присваивания копирования называется
Вызывается конструктор с плавающей запятой
Оператор присваивания копирования называется
Вызван деструктор
а 1234,43
б равно 10
с 42,4219
д равно 10
а равно 1234 как целое число
b равно 10 как целое число
с равно 42 как целое число
d равно 10 как целое число
Вызван деструктор
Вызван деструктор
Вызван деструктор
Вызван деструктор
$>
```

Глава VI

Упражнение 02: Теперь мы говорим

	Упражнение 02
Сейчас мы говорим	
Каталог сдачи: ex02/ Файлы	
для сдачи: Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp Разрешенные функции:	
roundf (из <cmath>)	

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

- 6 операторов сравнения: `>`, `<`, `>=`, `<=`, `==` и `!=`.
- 4 арифметических оператора: `+`, `-`, `*` и `/`.
- 4 оператора инкремента/декремента (преинкремент и постинкремент, преддекремент и постдекремент), которые будут увеличивать или уменьшать значение фиксированной точки от наименьшего представимого значения, такого как `1 + > 1`.

Добавьте эти четыре открытые перегруженные функции-члены в свой класс:

- Статическая функция-член `min`, принимающая в качестве параметров две ссылки на числа с фиксированной точкой и возвращающая ссылку на наименьшую из них.
- Статическая функция-член `min`, принимающая в качестве параметров две ссылки на константу числа 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 наибольшую из них.
- Статическая функция-член `max`, принимающая в качестве параметров две ссылки на константу числа с фиксированной точкой и возвращает ссылку на наибольшее из них.

Вы должны проверить каждую функцию вашего класса. Однако, запустив код ниже:

```
#include <iomanip>

ИНТ    основной (недействительный) {

    Фиксированный    а;
    Фиксированная константа    b(Фиксированный( 5.05f ) * Фиксированный( 2 ));

    std::cout << а << std::endl; std::cout << ++а
    << std::endl; std::cout << а << std::endl;
    std::cout << а++ << std::endl; std::cout << а <<
    std::endl;

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

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

    вернуть 0;
}
```


Должно выводить что-то вроде (для большей читабельности конструктор/деструктор mes мудрецы удалены в примере ниже):

```
$> ./a.out 0
0,00390625
0,00390625
0,00390625

0,0078125
10,1016
10.1016 $>
```

Глава VII

Упражнение 03: BSP

	Упражнение 03
НАПРИМЕР	
Каталог сдачи: ex03/	
Файлы для сдачи: Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp, Point.{h, hpp}, Point.cpp, bsp.cpp Разрешенные функции: roundf (из <cmath>)	

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

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

Очень полезно, не правда ли?



BSP означает разделение двоичного пространства. Пожалуйста. :)



Вы можете пройти этот модуль, не выполняя упражнение 03.

Начнем с создания класса Point в ортодоксальной канонической форме, представляющего двухмерную точку:

- Частные члены:

- Атрибут Fixed const x.
- Атрибут Fixed const y.
- Еще что-нибудь полезное.

- Общественные члены:

- Конструктор по умолчанию, который инициализирует x и y значением 0.
- Конструктор, принимающий в качестве параметров два постоянных числа с плавающей запятой. Он инициализирует x и y этими параметрами.
- Конструктор копирования.
- Перегрузка оператора присваивания копии.
- Деструктор.
- Еще что-нибудь 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, она вернет False.

Реализуйте и сдайте свои собственные тесты, чтобы убедиться, что ваш класс ведет себя так, как ожидалось.