C++ - Modül 02

Ad-hoc polymorphism, operator overloading ve Orthodox Canonical class formu

Özet: Bu doküman C++ modüllerinden Modül 02'nin alıştırmalarını içerir.
Versiyon: 8.2

İçindekiler

Bölüm I: Giriş

C++, Bjarne Stroustrup tarafından C programlama dilinin bir uzantısı olarak ya da "C with Classes" olarak oluşturulmuş genel amaçlı bir programlama dilidir (kaynak: Wikipedia).

Bu modüllerin amacı sizi Object-Oriented Programming'e (Nesne Yönelimli Programlama) tanıtmaktır. Bu, C++ yolculuğunuzun başlangıç noktası olacaktır. OOP öğrenmek için birçok dil önerilir, ancak eski dostunuz C'den türetildiği için C++'ı seçmeye karar verdik. Bu karmaşık bir dil olduğundan ve işleri basit tutmak için, kodunuz C++98 standardına uygun olacaktır.

Modern C++'ın birçok açıdan oldukça farklı olduğunun farkındayız. Bu nedenle, yetenekli bir C++ developer olmak istiyorsanız, 42 Common Core'dan sonra daha ileri gitmeniz size kalmıştır!

Bölüm II: Genel Kurallar

Derleme

- Kodunuzu c++ ve şu flaglerle derleyin: -Wall -Wextra -Werror
- -std=c++98 flagini eklediğinizde kodunuz hâlâ derlenmelidir

Biçimlendirme ve isimlendirme kuralları

- Alıştırma dizinleri şu şekilde adlandırılacaktır: ex00, ex01, ... , exn
- Dosyalarınızı, class'larınızı, function'larınızı, member function'larınızı ve attribute'larınızı kılavuzlarda belirtildiği şekilde adlandırın.
- Class isimlerini UpperCamelCase formatında yazın. Class kodu içeren dosyalar her zaman class ismine göre adlandırılacaktır. Örneğin: ClassName.hpp/ClassName.h, ClassName.cpp, veya ClassName.tpp. Bu durumda, bir tuğla duvarı temsil eden "BrickWall" class'ının tanımını içeren bir header dosyanız varsa, adı BrickWall.hpp olacaktır.
- Aksi belirtilmedikçe, her çıktı mesajı bir newline karakteriyle bitmeli ve standart çıktıya gösterilmelidir.
- Elveda Norminette! C++ modüllerinde herhangi bir kodlama stili zorlanmamaktadır. Favori stilinizi takip edebilirsiniz. Ancak akran değerlendiricilerinizin anlayamadığı kodun, not veremeyecekleri kod olduğunu unutmayın. Temiz ve okunabilir kod yazmak için elinizden geleni yapın.

İzin Verilen/Yasaklanan

Artık C'de kodlamıyorsunuz. C++ zamanı! Bu nedenle:

- Standard library'den neredeyse her şeyi kullanabilirsiniz. Bu yüzden, bildiğiniz şeylere bağlı kalmak yerine, mümkün olduğunca alıştığınız C function'larının C++'a özgü versiyonlarını kullanmak akıllıca olur.
- Ancak, başka hiçbir external library kullanamazsınız. Bu, C++11 (ve türetilmiş formları) ve Boost library'lerinin yasak olduğu anlamına gelir. Aşağıdaki function'lar da yasaktır: *printf(), *alloc() ve free(). Bunları kullanırsanız, notunuz 0 olur ve bu kadar.
- Açıkça aksi belirtilmedikçe, using namespace <ns_name> ve friend keyword'leri yasaktır. Aksi halde, notunuz -42 olur.
- STL'yi yalnızca Modül 08 ve 09'da kullanabilirsiniz. Bu şu anlama gelir: o zamana kadar Container'lar (vector/list/map ve benzeri) ve Algorithm'lar (<algorithm> header'ını include etmeyi gerektiren her şey) yasaktır. Aksi halde, notunuz -42 olur.

Birkaç tasarım gereksinimi

- C++'da da memory leak'leri oluşur. Memory ayırdığınızda (new keyword'ünü kullanarak), memory leak'lerinden kaçınmalısınız.
- Modül 02'den Modül 09'a kadar, açıkça aksi belirtilmedikçe class'larınız Orthodox Canonical Form'da tasarlanmalıdır.
- Bir header dosyasına konulan herhangi bir function implementasyonu (function template'leri hariç) alıştırma için 0 puan demektir.
- Her bir header'ınızı diğerlerinden bağımsız olarak kullanabilmelisiniz. Bu nedenle, ihtiyaç duydukları tüm dependency'leri include etmelidirler. Ancak, include guard'lar ekleyerek double inclusion problemini önlemelisiniz. Aksi halde, notunuz 0 olur.

Beni oku

- Gerekirse ek dosyalar ekleyebilirsiniz (örneğin, kodunuzu bölmek için). Bu ödevler bir program tarafından doğrulanmadığından, zorunlu dosyaları teslim ettiğiniz sürece bunu yapmaktan çekinmeyin.
- Bazen, bir alıştırmanın kılavuzları kısa görünebilir ancak örnekler, talimatlarda açıkça yazılmamış gereksinimleri gösterebilir.
- Başlamadan önce her modülü tamamen okuyun! Gerçekten, bunu yapın.
- Odin'e, Thor'a yemin olsun! Beyninizi kullanın!!!

C++ projeleri için Makefile konusunda, C'deki kuralların aynısı geçerlidir (Makefile hakkındaki Norm bölümüne bakın).

Birçok class implement etmeniz gerekecek. Bu sıkıcı görünebilir, tabii ki favori metin editörünüzü script yapabilme beceriniz yoksa.

Size alıştırmaları tamamlamak için belirli bir özgürlük verilmiştir. Ancak, zorunlu kurallara uyun ve tembel olmayın. Birçok yararlı bilgiyi kaçırırsınız! Teorik kavramlar hakkında okuma yapmaktan çekinmeyin.

Bölüm III: Yeni Kurallar

Bundan sonra, açıkça aksi belirtilmedikçe tüm class'larınız Orthodox Canonical Form'da tasarlanmalıdır. Bu durumda aşağıdaki dört gerekli member function'ı implement edeceklerdir:

- Default constructor
- Copy constructor
- Copy assignment operator
- Destructor

Class kodunuzu iki dosyaya bölün. Header dosyası (.hpp/.h) class tanımını içerirken, source dosyası (.cpp) implementasyonu içerir.

Bölüm IV: Alıştırma 00: Orthodox Canonical Formda İlk Class'ım

Alıştırma: 00
Orthodox Canonical Formda İlk Class'ım
Teslim dizini: ex00/
Teslim edilecek dosyalar: Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp
Yasak function'lar: Hiçbiri

Integer'ları ve floating-point sayıları bildiğinizi sanıyorsunuz. Ne kadar da şirin.

Lütfen bilmediğinizi keşfetmek için şu 3 sayfalık makaleyi (1, 2, 3) okuyun. Hadi, okuyun.

Bugüne kadar, kodunuzda kullandığınız her sayı temelde ya bir integer ya da floating-point sayıydı, ya da bunların herhangi bir varyantıydı (short, char, long, double ve benzeri).

Yukarıdaki makaleyi okuduktan sonra, integer'lar ve floating-point sayıların zıt karakteristiklere sahip olduğunu varsaymak güvenlidir.

Ama bugün, işler değişecek. Yeni ve harika bir sayı türünü keşfedeceksiniz: fixed-point sayılar! Çoğu dilin scalar türlerinde sonsuza kadar eksik olan fixed-point sayılar, performance, doğruluk, aralık ve precision arasında değerli bir denge sunar. Bu, fixed-point sayıların özellikle computer graphics, ses işleme veya bilimsel programlama gibi alanlarda uygulanabilir olmasını açıklar.

C++'da fixed-point sayılar bulunmadığından, bunları ekleyeceksiniz. Berkeley'den bu makale iyi bir başlangıçtır. Berkeley Üniversitesi'nin ne olduğu hakkında hiçbir fikriniz yoksa, Wikipedia sayfasının bu bölümünü okuyun.

Fixed-point sayıyı temsil eden Orthodox Canonical Form'da bir class oluşturun:

Private member'lar:

- Fixed-point sayı değerini depolamak için bir integer.
- Fractional bit sayısını depolamak için static constant integer. Değeri her zaman integer literal 8 olacaktır.

Public member'lar:

- Fixed-point sayı değerini 0'a başlatan default constructor.
- Copy constructor.
- Copy assignment operator overload.
- Destructor.
- Fixed-point değerinin ham değerini döndüren `int getRawBits( void ) const;` member function'ı.
- Fixed-point sayının ham değerini ayarlayan `void setRawBits( int const raw );` member function'ı.

Bu kodu çalıştırmak:

cpp

```cpp
#include <iostream>
int main( void ) {
  Fixed a;
  Fixed b( a );
  Fixed c;
  c = b;
  std::cout << a.getRawBits() << std::endl;
  std::cout << b.getRawBits() << std::endl;
  std::cout << c.getRawBits() << std::endl;
  return 0;
}
```

Şuna benzer bir çıktı vermelidir:

```
$> ./a.out
Default constructor called
Copy constructor called
Copy assignment operator called // <-- Bu satır implementasyonunuza bağlı olarak eksik olabilir
getRawBits member function called
Default constructor called
Copy assignment operator called
getRawBits member function called
getRawBits member function called
0
getRawBits member function called
0
getRawBits member function called
0
Destructor called
Destructor called
Destructor called
$>
```

Bölüm V: Alıştırma 01: Daha kullanışlı bir fixed-point number class'ına doğru

Alıştırma: 01
Daha kullanışlı bir fixed-point number class'ına doğru
Teslim dizini: ex01/
Teslim edilecek dosyalar: Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp
İzin verilen function'lar: roundf (<cmath>'den)

Önceki alıştırma iyi bir başlangıçtı, ancak class'ımız oldukça işe yaramaz. Yalnızca 0.0 değerini temsil edebiliyor.

Class'ınıza aşağıdaki public constructor'lar ve public member function'ları ekleyin:

- Parameter olarak constant integer alan constructor. Bunu karşılık gelen fixed-point değerine dönüştürür. Fractional bits değeri alıştırma 00'daki gibi 8'e initialize edilmelidir.
- Parameter olarak constant floating-point sayı alan constructor. Bunu karşılık gelen fixed-point değerine dönüştürür. Fractional bits değeri alıştırma 00'daki gibi 8'e initialize edilmelidir.
- Fixed-point değeri floating-point değere dönüştüren `float toFloat( void ) const;` member function'ı.
- Fixed-point değeri integer değere dönüştüren `int toInt( void ) const;` member function'ı.

Ve Fixed class dosyalarına aşağıdaki function'ı ekleyin:

- Fixed-point sayının floating-point temsilini parameter olarak geçilen output stream object'ine ekleyen insertion (<<) operator'ünün overload'ı.

Bu kodu çalıştırmak:

cpp

```cpp
#include <iostream>
int main( void ) {
  Fixed a;
  Fixed const b( 10 );
  Fixed const c( 42.42f );
  Fixed const 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() << " as integer" << std::endl;
  std::cout << "d is " << d.toInt() << " as integer" << std::endl;
  return 0;
}
```

Şuna benzer bir çıktı vermelidir:

```
$> ./a.out
Default constructor called
Int constructor called
Float constructor called
Copy constructor called
Copy assignment operator called
Float constructor called
Copy assignment operator called
Destructor called
a is 1234.43
b is 10
c is 42.4219
d is 10
a is 1234 as integer
b is 10 as integer
c is 42 as integer
d is 10 as integer
Destructor called
Destructor called
Destructor called
Destructor called
$>
```

Bölüm VI: Alıştırma 02: Şimdi konuşuyoruz

Alıştırma: 02
Şimdi konuşuyoruz
Teslim dizini: ex02/
Teslim edilecek dosyalar: Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp
İzin verilen function'lar: roundf (<cmath>'den)

Aşağıdaki operator'ları overload etmek için class'ınıza public member function'lar ekleyin:

- 6 comparison operator: >, <, >=, <=, ==, ve !=.
- 4 arithmetic operator: +, -, *, ve /.
- 4 increment/decrement (pre-increment ve post-increment, pre-decrement ve post-decrement) operator, bunlar fixed-point değeri en küçük temsil edilebilir ε ile artıracak veya azaltacak, öyle ki 1 + ε > 1.

Class'ınıza bu dört public overloaded member function'ı ekleyin:

- Parameter olarak fixed-point sayılara iki referans alan ve en küçüğüne referans döndüren static member function min.
- Parameter olarak constant fixed-point sayılara iki referans alan ve en küçüğüne referans döndüren static member function min.
- Parameter olarak fixed-point sayılara iki referans alan ve en büyüğüne referans döndüren static member function max.
- Parameter olarak constant fixed-point sayılara iki referans alan ve en büyüğüne referans döndüren static member function max.

Class'ınızın her özelliğini test etmek size kalmıştır. Ancak, aşağıdaki kodu çalıştırmak:

```cpp
#include <iostream>
int main( void ) {
  Fixed a;
  Fixed const b( Fixed( 5.05f ) * Fixed( 2 ) );
  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;
  return 0;
}
```

Şuna benzer bir şey çıktılamalıdır (daha iyi okunabilirlik için, aşağıdaki örnekte constructor/destructor mesajları kaldırılmıştır):

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

Eğer 0'a bölme işlemi yaparsanız, programın çökmesi kabul edilebilir.

Bölüm VII: Alıştırma 03: BSP

Alıştırma: 03
BSP
Teslim dizini: ex03/
Teslim edilecek dosyalar: Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp, Point.{h, hpp}, Point.cpp, bsp.cpp
İzin verilen function'lar: roundf (<cmath>'den)

Artık işlevsel bir Fixed class'ınız olduğuna göre, onu kullanmak güzel olurdu.

Bir noktanın üçgen içinde olup olmadığını belirten bir function implement edin. Çok yararlı, değil mi?

BSP, Binary Space Partitioning anlamına gelir. Rica ederiz. :)

Bu modülü alıştırma 03'ü tamamlamadan geçebilirsiniz.

2D nokta temsil eden Orthodox Canonical Form'da Point class'ını oluşturarak başlayalım:

Private member'lar:

- Fixed const attribute x.
- Fixed const attribute y.
- Diğer yararlı şeyler.

Public member'lar:

- x ve y'yi 0'a initialize eden default constructor.
- Parameter olarak iki constant floating-point sayı alan constructor. x ve y'yi bu parametrelerle initialize eder.
- Copy constructor.
- Copy assignment operator overload.
- Destructor.
- Diğer yararlı şeyler.

Sonuç olarak, uygun dosyada aşağıdaki function'ı implement edin:

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

- a, b, c: Sevgili üçgenimizin köşeleri.
- point: Kontrol edilecek nokta.
- Döndürür: Nokta üçgen içindeyse True. Aksi halde False.

Bu nedenle, nokta bir köşe ise veya bir kenar üzerindeyse, False döndürecektir.

Class'ınızın beklendiği gibi davrandığından emin olmak için kendi testlerinizi implement edin ve teslim edin.

Bölüm VIII: Teslim ve Akran Değerlendirmesi

Ödevinizi her zamanki gibi Git repository'nizde teslim edin. Savunma sırasında yalnızca repository'nizdeki çalışma değerlendirilecektir. Klasör ve dosya isimlerinizin doğru olduğundan emin olmak için iki kez kontrol etmekten çekinmeyin.

??????????? XXXXXXXXXX = 3d6f957a965f8361750a3ba6c97554e9f