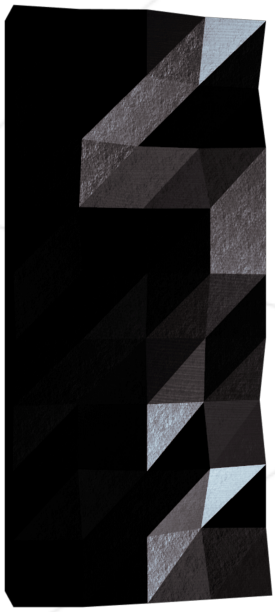

C++ - Modül 02

Geçici polimorfizm, operatör aşırı yüklemesi ve Ortodoks Kanonik sınıf formu

Özet:
Bu belge, C++ modüllerinden Modül 02'nin alıştırmalarını içerir.

Versiyon: 7

İçindekiler

Bölüm I

Giriş

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

Bu modüllerin amacı, sizi **Nesneye Dayalı Programlama** ile tanıştırmaktır. Bu, C++ yolculuğunuzun başlangıç noktası olacaktır. OOP öğrenmek için birçok dil önerilir. 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 çok farklı olduğunun farkındayız. Dolayısıyla, yetkin bir C++ geliştiricisi olmak istiyorsanız, 42 Ortak Çekirdekten sonra daha ileri gitmek size kalmış!

Bölüm II

Genel Kurallar

C++ modülleri için sadece C++98 kullanabilirsiniz. Amacınız nesne tabanlı programlamanın temellerini öğrenmek. Modern C++'ın bir çok yönden daha farklı olduğunu biliyoruz fakat C++'ta uzmanlaşmak için modern C++'a daha sonra ihtiyacınız olacak. Bu sizin C++ yolculuğunuzun başlangıç noktası olacak, Common Core'dan sonra ne kadar ilerleyeceğiniz size kalmış!

- Header'da implement edilen herhangi bir fonksiyon (templateler hariç), ve korunmayan headerlar 0 puan anlamına gelir.
- Tüm çıktılar standart çıktıya olacak (stdout) ve yeni satırla bitecek (aksi belirtilmediği takdirde).
- Belirtilen dosya adları harfi harfine takip edilmelidir. Class, fonksiyon ve metod isimleri dahil.
- Hatırlatma: Artık `C++`'da kodluyorsunuz, `C`'de değil Bu yüzden:
    - Belirtilen fonksiyonların kullanımı YASAKTIR, ve kullanımları `0` ile cezalandırılacaktır. : `*malloc`, `*printf` ve `free`.
    - Standart kütüphanedeki tüm fonksiyonlar serbesttir. ANCAK, sadece bildiklerinizi korumak yerine C'deki fonksiyonların C++'a uyarlanmış hallerini kullanmayı denemek akıllıca olur. Ne de olsa yeni bir dil öğreniyorsunuz. Ve HAYIR, STL kullanacağınız yer belirtilene kadar kullanamazsınız. (modül 08'e kadar) Yani vector/list/map vb. include <algorithm>'i gerektirecek hiçbir şeyi oraya gelene kadar kullanamazsınız.
- Aslına bakarsak, yasak olan herhangi bir fonksiyon ya da mekanizma direkt olarak `0` ile cezalandırılacaktır.
- Aksi belirtilmediği takdirde, `C++` keywordleri `"using namespace"` ve `"friend"` kullanımı yasaktır. Kullanımları direkt olarak `-42` ile cezalandırılacaktır.
- Classla alakalı dosyalar `ClassName.hpp` ve `ClassName.cpp` olacaktır (aksi belirtilmediği takdirde)
- Yükleme klasörleri: `ex00/`, `ex01/`, ..., `exn/`.

- Örnekleri iyice okumanız gerekmektedir. Egzersiz açıklamasında açıkça belli olmayan gereklilikleri içerebilirler.
- Başlangıçtan beri öğrendiğiniz `C++` araçlarını kullanmanız serbest olduğundan herhangi harici bir kütüphane kullanamazsınız. Siz sormadan, aynı zamanda `C++11` ve türevleri de bu yasağa dahil.
- Önemli sayıda class'ı teslim etmeniz gerekebilir. Kendi favori text editörünüzü scriptleyene kadar bu sıkıcı görünebilir.
- Başlamadan önce tüm egzersizleri TAMAMEN okuyun! Yapın bunu.
- Kullanılacak derleyici: `c++`.
- Kodlarınız şu bayraklarla derlenecektir. `-Wall -Wextra -Werror`.
- Tüm include'lar diğerleri tarafından bağımsızca include edilebilmelidir. Include'lar bağlı oldukları diğer include'ları içermelidir.
- Merak ediyorsanız, belirli herhangi bir kodlama stili istemiyoruz. İstediğiniz stilde yazabilirsiniz, kısıtlama yok. Şunu unutmayın ki sizi değerlendirecek arkadaşlarınız kodu okuyamazsa, notlandıramazlar da.
- Önemli bir şey: Özellikle belirtilmediği sürece, sizi bir program kontrol etmeyecek. Bu yüzden, egzersizleri istediğiniz şekilde yapmakta özgürsünüz. Ancak, her egzersizin kısıtlamasına dikkat edin, ve tembel OLMAYIN, sundukları BİRÇOK şeyi kaçırabilirsin.
- Yüklediğiniz dosyalarda bazı ekstra dosyalar olması sorun değil, , kodunuzu sizden istenenden daha fazla parçaya ayırabilirsiniz. Sonucunuz program tarafından belirlenmeyeceği sürece bu konuda özgür hissedebilirsiniz.
- Egzersiz açıklama dosyası kısa bile olsa sizden tam olarak ne beklendiğini anlamak için zaman harcayıp egzersizi mümkün olan en iyi şekilde yapmanıza değer.
- Odin'den, Thor'dan! Beynini kullan!!!

Bölüm III

Yeni kurallar

Şu andan itibaren, aksi açıkça belirtilmedikçe, tüm sınıflarınız **Ortodoks Kanonik Form**'da tasarlanmalıdır. Ardından, aşağıdaki dört gerekli üye işlevini uygulayacaklar:

- Varsayılan yapıcı
- Kopyalama Kurucusu
- Atama operatörünü kopyalama
- Yıkıcı

Split your class code into two files. The header file (.hpp/.h) contains the class definition whereas the source file (.cpp) contains the implementation.

Bölüm IV

Alıştırma 00: Ortodoks Kanonik Formda Birinci Sınıfım

	Exercise : 00
Ortodoks Kanonik Formda Birinci Sınıfım	
Turn-in directory : *ex*00/	
Files to turn in : `Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp`	
Forbidden functions : `None`	

Integerları ve kayan noktalı sayıları bildiğinizi sanıyorsunuz. Ne kadar tatlı.

Lütfen bilmediğinizi keşfetmek için bu 3 sayfalık makaleyi okuyun (1, 2, 3) Devam edip, onları oku.

Bugüne kadar, kodunuzda kullandığınız her sayı, temelde ya bir tam sayı ya da bir kayan noktalı sayı ya da bunların herhangi bir türeviydi (`short`, `char`, `long`, `double`). ve benzeri). Yukarıdaki makaleyi okuduktan sonra, tam sayıların ve kayan noktalı sayıların zıt özelliklere sahip olduğunu varsaymak güvenlidir.

Ama bugün işler değişecek. Yeni ve harika bir sayı türü keşfedeceksiniz: **sabit noktalı sayılar**! Çoğu dilin skaler türünden her zaman eksik olan sabit noktalı sayılar, performans, doğruluk, aralık ve kesinlik arasında değerli bir denge sunar. Bu, sabit noktalı sayıların bilgisayar grafiklerine, ses işlemeye veya bilimsel programlamaya neden özellikle uygulanabilir olduğunu açıklar.

C++ sabit noktalı sayılardan yoksun olduğundan, onları ekleyeceksiniz. Bu makale Berkeley'den iyi bir başlangıç. Berkeley Üniversitesi'nin ne olduğu hakkında hiçbir fikriniz yoksa, Wikipedia sayfasının bu bölümünü okuyun.

Sabit noktalı bir sayıyı temsil eden Ortodoks Kanonik Formunda bir sınıf oluşturun:

- Özel üyeler:
  - Sabit noktalı sayı değerini depolamak için bir **integer**.
  - Kesirli bitlerin sayısını depolamak için bir **statik sabit tamsayı**. Değeri her zaman tamsayı `8` olacaktır.
- Genel üyeler:
  - Sabit nokta sayı değerini `0` olarak başlatan varsayılan bir kurucu
  - Bir kopyalama kurucusu
  - Bir kopyalama atama operatörünü aşırı yükleme
  - Bir yıkıcı
  - Bir üye işlevi `int getRawBits( void ) const;` bu, sabit nokta değerinin ham değerini döndürür.
  - Sabit noktalı sayının ham değerini ayarlayan `void setRawBits( int const raw );` üye işlevi.

Bu kodu çalıştırmak:

```
#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 // <-- This line may be missing depending on your implementation
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 sabit noktalı sayı sınıfına doğru

	Exercise 01
Daha kullanışlı bir sabit noktalı sayı sınıfına doğru	
Turn-in directory : *ex*01/	
Files to turn in : `Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp`	
Allowed functions : `roundf (from <cmath>)`	

Önceki alıştırma iyi bir başlangıçtı ama sınıfımız oldukça işe yaramaz. O yalnızca `0.0` değerini temsil edebilir.

Aşağıdaki genel oluşturucuları ve genel üye işlevlerini sınıfınıza ekleyin:

- Parametre olarak bir **sabit tamsayı** alan bir yapıcı.
  Bunu karşılık gelen sabit nokta değerine dönüştürür. Kesirli bit değeri, alıştırma 00'daki gibi 8'e başlatılır.
- Parametre olarak bir **sabit kayan noktalı sayı** alan bir yapıcı.
  Bunu karşılık gelen sabit nokta değerine dönüştürür. Kesirli bit değeri, alıştırma 00'daki gibi 8'e başlatılır.
- Bir üye işlevi `float toFloat( void ) const;`
  sabit noktalı değeri kayan noktalı bir değere dönüştürür.
- Sabit nokta değerini bir tamsayı değerine dönüştüren `int toInt( void ) const;` üye işlevi.

Ve aşağıdaki işlevi **Sabit** sınıf dosyalarına ekleyin:

- Sabit noktalı sayının kayan noktalı gösterimini parametre olarak iletilen çıkış akışı nesnesine ekleyen ekleme («) operatörünün aşırı yüklenmesi.

Bu kodu çalıştırmak:

```
#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

	Exercise 02
Şimdi konuşuyoruz	
Turn-in directory : $ex02/$	
Files to turn in : `Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp`	
Allowed functions : `roundf (from <cmath>)`	

Aşağıdaki operatörleri aşırı yüklemek için sınıfınıza genel üye işlevleri ekleyin:

- 6 karşılaştırma operatörü: `>`, `<`, `>=`, `<=`, `==` ve `!=`.
- 4 aritmetik operatör: `+`, `-`, `*` ve `/`.
- $1 + \epsilon > 1$ gibi temsil edilebilir en küçük ϵ'dan sabit noktalı değeri artıracak veya azaltacak 4 artırma/azaltma (arttırma öncesi ve artırma sonrası, ön eksiltme ve eksiltme sonrası) operatörleri.

Bu dört genel aşırı yüklenmiş üye işlevini sınıfınıza ekleyin:

- Sabit noktalı sayılarda iki referansı parametre olarak alan ve en küçüğüne bir referans döndüren bir statik üye işlevi `min`.
- **constant** sabit noktalı sayılara iki başvuruyu parametre olarak alan ve en küçüğüne bir başvuru döndüren bir statik üye işlevi `min`.
- Sabit noktalı sayılarda iki referansı parametre olarak alan ve en büyüğüne bir referans döndüren bir statik üye işlevi `max`.
- **constant** sabit noktalı sayılara iki başvuruyu parametre olarak alan ve en büyüğüne bir başvuru döndüren bir statik üye işlevi `max`.

Sınıfınızın her özelliğini test etmek size kalmış. Yine de, aşağıdaki kodu çalıştırabilirsin:

```
#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 çıktı vermeli (daha fazla okunabilirlik için, aşağıdaki örnekte yapıcı/yıkıcı mesajları kaldırılmıştır):

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

Bölüm VII

Alıştırma 03: BSP

	Exercise 03
BSP	
Turn-in directory : *ex*03/	
Files to turn in : `Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp, Point.{h, hpp}, Point.cpp, bsp.cpp`	
Allowed functions : `roundf (from <cmath>)`	

Artık işlevsel bir **Sabit** sınıfınız olduğuna göre, onu kullanmak güzel olurdu.

Bir noktanın üçgenin içinde olup olmadığını gösteren bir fonksiyon uygulayın. Çok faydalı, değil mi?

`BSP, Binary alan bölümleme anlamına gelir. Rica ederim. :)`

`Bu modülü egzersiz 03 yapmadan geçebilirsiniz.`

2B noktayı temsil eden Ortodoks Kanonik Formda **Nokta** sınıfını oluşturarak başlayalım:

- Özel üyeler:
  - Sabit bir const öznitelik `x`.
  - Sabit bir const öznitelik `y`.
  - İşe yarayabilecek her şey.
- Genel üyeler:
  - `x` ve `y`'yi `0` olarak başlatan varsayılan bir kurucu.
  - İki sabit kayan noktalı sayıyı parametre olarak alan bir yapıcı. Bu parametrelerle `x` ve `y`'yi başlatır.
  - Bir Kopyalama Kurucusu
  - Bir kopyalama atama operatörünü aşırı yükleme.
  - Bir yıkıcı.
  - İşe yarayabilecek her şey.

Sonuç olarak, uygun dosyada aşağıdaki işlevi uygulayın:

```
bool bsp( Nokta const a, Nokta const b, Nokta const c, Nokta const nokta);
```

- a, b, c: Sevgili üçgenimizin köşeleri.
- nokta: Kontrol edilecek nokta.
- Returns: Nokta üçgenin içindeyse True. Aksi halde False.
  Böylece, nokta bir tepe noktası veya kenardaysa False değerini döndürür.

Sınıfınızın beklendiği gibi davrandığından emin olmak için kendi testlerinizi uygulayın ve teslim edin.