

C++ - Modül 04

Alt tip polimorfizmi, soyut sınıflar, arayüzler

Özet: Bu belge, C++ modüllerinden Modül 04'ün alıştırmaları içerir.

Sürüm: 10

İçindekiler

BEN	giriş	2
II	Genel kurallar	3
III	Alı ştı rma 00: Polimorfizm	5
IV	Alı ştı rma 01: Dünyayı ateşe vermek istemiyorum	7
V	Alı ştı rma 02: Soyut sı nı f	9
VI	Alı ştı rma 03: Arayüz ve özet	10

Bölüm I

giriş

C++, Bjarne Stroustrup tarafı ndan C programlama dilinin veya "C with Classes"ı n (kaynak: [Wikipedia](#)) bir uzantı sı olarak oluşturulan genel amaçlı bir programlama dilidir .

Bu modüllerin amacı sizi Nesneye Dayalı Programlamaya tanı tmaktır .
Bu, C++ yolculuğunuzun başlangı ç noktası olacaktır r. OOP öğrenmek için birçok dil önerilmektedir. Eski dostunuz C'den türetildiği için C++'ı seçmeye karar verdik.
Bu karmaşı k bir dil olduğundan ve işleri basitleştirmek için kodunuz C++98 standardı na uygun olacaktır r.

Modern C++'ı n birçok açı dan çok farklı olduğunun farkı ndayız. Dolayısı yla yetkin bir C++ geliştiricisi olmak istiyorsanız 42 Ortak Çekirdekten sonra daha da ileri gitmek size kalmı ş!

Bölüm II

Genel kurallar

Derleme

- Kodunuzu c++ ve -Wall -Wextra -Werror işaretleriyle derleyin
- -std=c++98 bayrağı nı eklerseniz kodunuz yine de derlenmelidir.

Biçimlendirme ve adlandırma kuralları

- Alıştı rma dizinleri şu şekilde adlandırılacaktır: ex00, ex01, ... , eski
- Dosyaları nı zı , sı nı fları nı zı , işlevlerinizi, üye işlevlerinizi ve niteliklerinizi gerektiği gibi adlandırın. yardımcı notlar.
- Sı nı f adları nı UpperCamelCase formatı nda yazın . Sı nı f kodunu içeren dosyalar her zaman sı nı f adı na göre adlandırılmalıdır. Örneğin: ClassName.hpp/ClassName.h, ClassName.cpp veya ClassName.tpp. Daha sonra, tuğla duvarı temsil eden "BrickWall" sı nı fı nı n tanı mı nı içeren bir başlık dosyası z varsa, adı BrickWall.hpp olacaktır .
- Aksi belirtilmediği sürece her çıktı ş mesajı yeni satı rla sonlandırılmalıdır. karakter ve standart çıktı da görüntülenir.
- Güle güle Norminette! C++ modüllerinde hiçbir kodlama stili uygulanmaz. En sevdiğiniz kişiyi takip edebilirsiniz. Ancak akran değerlendiricilerinizin anlayamadığı bir kodun, not veremeyecekleri bir kod olduğunu unutmayın. Temiz ve okunabilir bir kod yazmak için elinizden geleni yapın .

İzin verildi/Yasaklandı

Artık C'de kod yazmı yorsunuz. C++ zamanı ! Öyleyse:

- Standart kütüphanedeki hemen hemen her şeyi kullanmanızı izin verilmektedir. Bu nedenle, zaten bildiklerinize bağlı kalmak yerine, alıştı ğı nı z C fonksiyonları nı n mümkün olduğunca C++ benzeri versiyonları nı kullanmak akı llı ca olacaktır .
- Ancak başka herhangi bir harici kütüphaneyi kullanamazsınız . Bu, C++11'in (ve türetilmiş formları nı) ve Boost kitaplı kları nı n yasak olduğu anlamı na gelir. Şu işlevler de yasaktır: *printf(), *alloc() ve free(). Bunları kullanı rsanız notunuz 0 olacaktır , hepsi bu.

- Aksi açıkça belirtilmediği sürece, kullanılan ad alanı `<ns_name>` ve arkadaş anahtar kelimeleri yasaktır. Aksi takdirde notunuz -42 olacaktır.
- STL'yi yalnızca Modül 08 ve 09'da kullanmanıza izin verilmektedir. Bunun anlamı : O zamana kadar Konteyner yok (vektör/liste/harita/vb.) ve Algoritma yok (`<algorithm>` başlığı dahil etmeyi gerektiren herhangi bir şey). Aksi takdirde notunuz -42 olacaktır.

Birkaç tasarım gereksinimi

- C++'ta da bellek sızıntısı yaşanıyor. Bellek ayırdığınızda (yeni belleği kullanarak) anahtar kelime), bellek sızıntıları ndan kaçınmalısınız.
- Modül 02'den Modül 09'a kadar sınıflarınızı Ortodoks tarzında tasarlanmalıdır. Kanonik Form, aksi açıkça belirtilmediği sürece.
- Bir başlık dosyasına yerleştirilen herhangi bir fonksiyon uygulaması (fonksiyon şablonları hariç), alıştırma için 0 anlamına gelir.
- Başlıklarınızı diğerlerinden bağımsız olarak kullanabilmelisiniz. Bu nedenle ihtiyaç duydukları tüm bağımlılıkları içermelidirler. Ancak, içerme korumaları ekleyerek çift dahil etme probleminden kaçınmalısınız. Aksi takdirde notunuz 0 olacaktır.

Beni oku

- Gerekirse (örn. kodunuzu bölmek için) bazı ek dosyalar ekleyebilirsiniz. Bu ödevler bir program tarafı ndan doğrulanmadığından, zorunlu dosyaları teslim ettiğiniz sürece bunu yapmaktan çekinmeyin.
- Bazen bir alıştırmanın yönergeleri kısıtlı görünebilir ancak örnekler bunu gösterebilir. Talimatlarda açıkça yazılmayan gereksinimler.
- Başlamadan önce her modülü tamamen okuyun! Gerçekten yap.
- Odin adını, Thor adını! Beynini kullan!!!




Çok sayıda dersi uygulamayı gerekecek. Bu sıkıcı görünebilir, favori metin düzenleyicinizin komut dosyasını yazamadığınız sürece.



Egzersizleri tamamlamanız için size belirli bir miktar özgürlük verilir. Ancak zorunlu kurallara uyun ve tembel olmayın. Yapabilirdin birçok yararlı bilgiyi kaçırsınız! Hakkında okumaktan çekinmeyin teorik kavramlar.

Bölüm III

Alıştırma 00: Polimorfizm

	Egzersiz : 00
Polimorfizm	
Teslim dizini: ex00/ Teslim	
edilecek dosyalar: Makefile, main.cpp, *.cpp, *.h, *.hpp	
Yasaklanan işlevler : Yok	

Her alıştırma için yapabileceğiniz en eksiksiz testleri sunmalı sı nı z .
Her sı nı f n yapı cı ları ve yı k cı ları belirli mesajları görüntülemelidir. Tüm sı nı flar için aynı mesajı kullanmayı n.

Animal adı nda basit bir temel sı nı f uygulayarak başlayı n . Korumalı biri var bağlanmak:

- std::string türü;

Animal'dan miras alan bir Dog sı nı fı uygulayı n.

Animal'dan miras alan bir Cat sı nı fı uygulayı n.

Bu iki türetilmiş sı nı f, tür alanları nı adları na göre ayarlamalı dı r. Daha sonra, Köpeğin türü "Köpek" olarak başlatı lacak ve Kedinin türü "Kedi" olarak başlatı lacaktı r.
Animal sı nı fı nı n türü boş bı rakı labilir veya istediğiniz değere ayarlanabilir.

Her hayvan üye fonksiyonunu kullanabilmelidir: makeSound()

Uygun bir ses çı karacaktır (kediler havlamaz).

Bu kodu çalıştırmak, Hayvan sınıflarını değil, Köpek ve Kedi sınıflarını belirli seslerini yazdıracaktı r.


```
int ana() {  
  
    const Hayvan* meta = new Hayvan(); const  
    Hayvan* j = new Köpek(); const Animal*  
    i = new Cat();  
  
    std::cout << j->getType() << << std::endl; std::cout << i->getType() <<  
    << std::endl; i->makeSound(); //kedi sesini çıkaracağı z! j->  
    >makeSound(); meta->makeSound();  
  
    ...  
  
    değerini döndür ;  
}
```

Nasıl çalıştırıldığıni anladığınızı zdan emin olmak için WrongAnimal sınıfından miras alan bir WrongCat sınıfını uygulayın . Yukarıdaki kodda Animal ve Cat'i yanlı ş olanlarla değiştirirseniz, WrongCat WrongAnimal sesini çıkarırmalıdır .

Yukarıda verilenlerden daha fazla test uygulayın ve teslim edin.

Bölüm IV

Egzersiz 01: Dünyayı ateşe vermek istemiyorum

	Egzersiz : 01
Dünyayı ateşe vermek istemiyorum	
Teslim dizini : ex01/ Teslim	
edilecek dosyalar : Önceki alışı rmadan dosyalar + *.cpp, *.h, *.hpp}	
Yasaklanan işlevler : Yok	

Her sınıfın fonksiyonları ve yapıları belirli mesajları görüntülemelidir.

Bir Beyin sınıfı uygulayın. Fikirler adı verilen 100 std::string dizisi içerir. Bu şekilde Köpek ve Kedinin özel bir Beyin* özelliği olacaktır. İnşaatı ardından Köpek ve Kedi, yeni Beyin() kullanarak Beyinlerini yaratacak; Yıkılma üzerine Köpek ve Kedi Beyinlerini silecektir.

Ana işlevinizde bir dizi Animal nesnesi oluşturun ve doldurun. Yarıısı Dog nesneleri, diğer yarıısı ise Cat nesneleri olacak. Programın sonu yürütülmesinin sonunda, bu dizi üzerinde döngü yapıp ve her Hayvanı silin. Hayvan olarak köpekleri ve kedileri doğrudan silmelisiniz. Uygun yıkılma beklenen sırayla çağrılmalıdır.

Bellek sızıntılarını kontrol etmeyi unutmayın.


Bir Köpeğin veya Kedinin kopyası sağlanmamalıdır. Bu nedenle kopyaları oluşturmayın ve derin kopya olup olmadığını test etmelisiniz!


```
int ana() {  
  
    const Hayvan* j = new Köpek(); const  
    Animal* i = new Cat();  
  
    sil ;//bir sınıfı yaratmamalı delete i;  
  
    ...  
  
    değerini döndür ;  
}
```

Yukarı da verilenlerden daha fazla test uygulayı n ve teslim edin.

Bölüm V

Alıştırma 02: Soyut sınıf

	Egzersiz : 02
Soyut sınıf	
Teslim edilecek dizin : ex02/	
Teslim edilecek dosyalar : Önceki alıştırma dosyaları + *.cpp, *.h, *.hpp	
Yasaklanan işlevler : Yok	


Hayvan nesneleri yaratmak sonuçta mantıklı değil. Doğrudur, ses çıkmazlar!

Olası hataları önlemek için varsayılan `Animal` sınıfını somutlaştırmayı labilir olmaması gerekir. `Animal` sınıfını kimsenin örnekleyemeyeceği şekilde düzeltin. Her şey eskisi gibi çalışmalı.

İsterseniz `Animal`'a A öneki ekleyerek sınıf adını güncelleyebilirsiniz.

Bölüm VI

Alıştırma 03: Arayüz ve özet

	Egzersiz : 03
Arayüz ve özet	
Teslim dizini: ex03/ Teslim	
edilecek dosyalar: Makefile, main.cpp, *.cpp, *.h, *.hpp	
Yasaklanan işlevler : Yok	

Arayüzler C++98'de mevcut değildir (C++20'de bile). Ancak saf soyut sınıflara genellikle arayüzler denir. Bu nedenle, bu son alıştırma da, bu modülü aldığınızdan emin olmak için arayüzleri uygulamaya çalışalım.

Aşağıdaki AMateria sınıfını tanımlayın ve gerekli üye işlevlerini uygulayın.

```
A sınıfı Malzeme
{
    korumalı : [...]

public:
    AMateria(std::string const & type); [...]

    std::string const & getType() const; //Malzeme türünü döndürür

    sanal AMateria* clone() const = 0; sanal geçersiz
    kullanımı (ICharacter& hedef);

};
```

Materia'yı Ice ve Cure beton sınıflarıyla uygulayın. Türlerini ayarlamak için adları küçük harflerle kullanın (Buz için "buz", Cure için "tedavi"). Elbette üye işlevleri clone() aynı türden yeni bir örnek döndürecektir (yani, bir Ice Materia'yı klonlarsanız yeni bir Ice Materia elde edersiniz).

use(ICharacter&) üye işlevi şunu gösterecektir:

- Buz: "* <isim>'e buz oku atar *"
- Tedavi: "* <isim>'in yaraları iyileştirir *"

<name> parametre olarak iletilen Karakterin adıdır. Açık ayraçları (< ve >) yazdırılmayın.



Bir Materia'yı diğerine atarken türü kopyalamak
algoritması.

Aşağıdaki arayüzü uygulayacak somut sınıf Karakterini yazın:

```
sınıf I Karakter {

    public:
        virtual ~ICharacter() {} virtual
        std::string const & getName() const = 0; sanal geçersiz
        donatı m(AMateria* m) = 0; sanal void
        unequip(int idx) = 0; sanal geçersiz kullanı m
        (int idx, ICharacter& target) = 0;

};
```

Karakterin 4 slotluk bir envanteri vardır, bu da en fazla 4 Materia anlamına gelir. Envanter inşaat aşamasında boştur. Buldukları ilk boş yuvaya Materia'ları yerleştirirler. Bu şu anlama gelir: slot 0'dan slot 3'e. Tam bir envantere Materia eklemeye çalışırlarsa veya mevcut olmayan bir Materia'yı kullanmaya/donatmayı kaldırmaya çalışırlarsa, hiçbir şey yapmayı n (ancak yine de hatalar yasaktır). unequip() üye işlevi Materia'yı SİLMEKELİDİR!



Karakterinizin yerde bıraktığı Materia'ları dilediğiniz gibi kullanın.
unequip() işlevini veya başka bir şeyi çağırmadan önce adresleri kaydedin, ancak bellek sızıntıları ndan kaçınmanızı gerektirir unutmayın.

use(int, ICharacter&) üye fonksiyonunun Materia'yı kullanması gerekecektir. slot[idx] ve hedef parametreyi AMateria::use işlevine iletin.



Karakterinizin envanteri her türlü şeyi destekleyebilecektir.
AMateria.

Karakterinizin adı nı parametre olarak alan bir kurucuya sahip olması gerekir . Bir Karakterin herhangi bir kopyası (kopya oluşturucu veya kopya atama işlemi kullanılarak) derin olmalıdır . Kopyalama sırasında, yenilerinin envantere eklenmesinden önce bir Karakterin Materias'ı nı n silinmesi gerekir. Elbette, bir Karakter yok edildiğinde Materyallerin silinmesi gerekir.

Aşağıdaki arayüzü uygulayacak somut MateriaSource sınıfını yazın :

```
sınıf IMateriaSource
{
    public:
        virtual ~IMateriaSource() {} virtual
        void LearnMateria(AMateria*) = 0; sanal AMateria*
        createMateria(std::string const & type) = 0;
};
```

- LearnMateria(AMateria*)

Parametre olarak iletilen Materia'yı kopyalar ve daha sonra klonlanabilmesi için hafızada saklar. Karakter gibi, MateriaSource da en fazla 4 Materia'yı bilebilir. Mutlaka benzersiz olmaları gerekmez.

- createMateria(std::string const &)

Yeni bir Materia döndürür. İkinci , türü parametre olarak iletilen türe eşit olan MateriaSource tarafı ndan daha önce öğrenilen Materia'nı n bir kopyasıdır . Tür bilinmiyorsa 0 değerini döndürür.

Özetle, MateriaSource'unuz gerektiğinde Materias'ı n "şablonları nı " oluşturabilmek için öğrenebilmelidir. Daha sonra, yalnızca türünü tanımlayan bir dize kullanarak yeni bir Materia oluşturabileceksiniz.

Bu kodu çalıştırıyorum yorum:

```
int ana() {  
  
    IMateriaSource* src = new MateriaSource(); src->learnMateria(new  
    Ice()); src->learnMateria(new Cure());  
  
    ICharacter* me = new Character("ben");  
  
    AMateria* tmp; tmp  
    = src->createMateria("buz"); ben->donat(tmp);  
    tmp = src-  
    >createMateria("tedavi"); ben->donat(tmp);  
  
    ICharacter* bob = new Character("bob");  
  
    ben->kullan(0, *bob); ben-  
    >kullan(1, *bob);  
  
    sil ; beni  
    Sil; kaynağı  
    sil;  
  
    değeri döndür;  
}
```

Çıktı vermeli:

```
$> clang++ -W -Wall -Werror *.cpp $> ./a.out | cat  
-e * bob'a buz oku atar *$ *  
bob'un yaraları nı iyileştirir *$
```

Her zamanki gibi yukarı da verilenlerden daha fazla test uygulayın ve teslim edin.



Bu modülü egzersiz 03'ü yapmadan geçebilirsiniz.