



C++ - Modül 06

C++ dökümleri

Özet:

Bu doküman C++ modüllerinden Modül 06'nın alıştırma ve örneklerini içermektedir.

Sürüm: 5.1

İçindekiler

I	Giriş	2
II	Genel kurallar	3
III	Ek kural	5
IV	Alıştırma 00: Skaler tiplerin dönüştürülmesi	6
V	Alıştırma 01: Serileştirme	9
VI	Alıştırma 02: Gerçek türü tanımlama	10

Bölüm I Giriş

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

Bu modüllerin amacı sizi **Nesne Yönelimli Programlama** ile tanıştırmaktır. Bu, C++ yolculuğunuzun başlangıç noktası olacaktır. OOP öğrenmek için birçok dil önerilmektedir. 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 pek çok açıdan çok farklı olduğunun farkındayız. Dolayısıyla, yetkin bir C++ geliştiricisi olmak istiyorsanız, 42 Common Core'dan sonra daha ileri gitmek size kalmış!

Bölüm II Genel

kurallar

Derleme

- Kodunuzu c++ ve -Wall -Wextra -Werror bayrakları ile derleyin
- Eğer -std=c++98 bayrağını eklerseniz kodunuz yine de derlenmelidir

Biçimlendirme ve adlandırma kuralları

- Alıştırma dizinleri şu şekilde adlandırılacaktır: ex00, ex01, ... , exn
- Dosyalarınızı, sınıflarınızı, işlevlerinizi, üye işlevlerinizi ve niteliklerinizi yönergelerde belirtildiği şekilde adlandırın.
- Sınıf adlarını **UpperCamelCase** biçiminde yazın. Sınıf kodu içeren dosyalar her zaman sınıf adına göre adlandırılacaktır. Örneğin: ClassName.hpp/ClassName.h, ClassName.cpp veya ClassName.tpp. Bu durumda, tuğla duvar anlamına gelen "BrickWall" sınıfının tanımını içeren bir başlık dosyanız varsa, adı BrickWall.hpp olacaktır.
- Aksi belirtilmedikçe, her çıktı mesajı bir yeni satır karakteriyle sonlandırılmalı ve standart çıktı da görüntülenmelidir.
- *Güle güle Norminette!* C++ modüllerinde herhangi bir kodlama stili zorunlu değildir. En sevdiğiniz takip edebilirsiniz. Ancak, akran değerlendiricilerinizin anlayamadığı bir kodun not veremeyecekleri bir kod olduğunu unutmayın. Temiz ve okunabilir bir kod yazmak için elinizden geleni yapın.

İzinli/Yasak

Artık C'de kodlama yapmıyorsunuz. C++ zamanı! Bu nedenle:

- Standart kütüphanedeki neredeyse her şeyi kullanmanıza izin verilir. Bu nedenle, zaten bildiklerinize bağlı kalmak yerine, alışkın olduğunuz C işlevlerinin C++-ish sürümlerini mümkün olduğunca kullanmak akıllıca olacaktır.
- Ancak, başka herhangi bir harici kütüphane kullanamazsınız. Bu, C++11 (ve türetilmiş formlar) ve Boost kütüphanelerinin yasak olduğu anlamına gelir. Aşağıdaki fonksiyonlar da yasaktır: *printf(), *alloc() ve free(). Eğer bunları kullanırsanız, notunuz 0 olacaktır ve hepsi bu kadar.

- Aksi açıkça belirtilmedikçe, using ad alanının <ns_name> ve arkadaş anahtar kelimeleri yasaktır. Aksi takdirde notunuz -42 olacaktır.
- **STL'yi sadece Modül 08 ve 09'da kullanmanıza izin verilmektedir.** Bunun anlamı: o zamana kadar **Kapsayıcı** (vektör/liste/harita/ve benzeri) ve **Algoritma** (<algorithm> başlığını içermesi gereken herhangi bir şey) kullanmayacaksınız. Aksi takdirde notunuz -42 olacaktır.

Birkaç tasarım gereksinimi

- C++'da da bellek sızıntısı meydana gelir. Bellek ayırdığınızda (new anahtar sözcüğü), **bellek sızıntılarından** kaçınmalısınız.
- Modül 02'den Modül 09'a kadar, **aksi açıkça belirtilmediği sürece**, dersleriniz **Ortodoks Kanonik Formunda** tasarlanmalıdır.
- Bir başlık dosyasına konulan herhangi bir işlev uygulaması (işlev şablonları hariç) alıştırma için 0 anlamına gelir.
- Başlıklarınızın her birini diğerlerinden bağımsız olarak kullanabilmelisiniz. Bu nedenle, ihtiyaç duydukları tüm bağımlılıkları içermelidirler. Ancak, **include korumaları** ekleyerek çifte içerme sorunundan kaçınmalısınız. Aksi takdirde notunuz 0 olacaktır.

Beni okuyun

- Gerekirse bazı ek dosyalar ekleyebilirsiniz (örneğin, kodunuzu bölmek için). Bu ödevler bir program tarafından doğrulanmadığından, zorunlu dosyaları teslim ettiğiniz sürece bunu yapmaktan çekinmeyin.
- Bazen bir alıştırmının yönergeleri kısa görünebilir ancak örnekler, yönergelerde açıkça yazılmayan gereksinimleri gösterebilir.
- Başlamadan önce her modülü tamamen okuyun! Gerçekten okuyun.
- Odin tarafından, Thor tarafından! Beynini kullan!!!



Çok sayıda sınıf uygulamanız gerekecek. En sevdiğiniz metin düzenleyicinizi komut dosyası haline getiremediğiniz sürece bu sıkıcı görünebilir.



Egzersizleri tamamlamanız için size belirli bir miktar özgürlük verilir. Ancak, zorunlu kurallara uyun ve tembellik etmeyin. Pek çok faydalı bilgiyi kaçırsınız! Teorik kavramlar hakkında okumaktan çekinmeyin.

Bölüm III Ek


kural

Aşağıdaki kural tüm modül için geçerlidir ve isteğe bağlı değildir.

Her alıştırma için, tür dönüşümü belirli bir döküm türü kullanılarak çözülmelidir.
Seçiminiz savunma sırasında kontrol edilecektir.

Bölüm IV

Alıştırma 00: Skaler tiplerin dönüştürülmesi

	Egzersiz 00
Skaler tiplerin dönüştürülmesi	
Giriş dizini : <i>ex00/</i>	
Teslim edilecek dosyalar : Makefile, *.cpp, *.{h, hpp}	
İzin verilen fonksiyonlar : Bir dizeden int, float veya double'a dönüştürmek için herhangi bir fonksiyon. Bu yardımcı olacaktır, ancak tüm işi yapmayacaktır.	

Parametre olarak en yaygın biçimiyle bir C++ değişmezinin dize gösterimini alan bir "convert" yöntemi içerecek statik bir ScalarConverter sınıfı yazın. Bu değişmez aşağıdaki skaler tiplerden birine ait olmalıdır:

- char
- int
- şamandıra
- çift

Karakter parametreleri dışında, yalnızca ondalık gösterim kullanılacaktır.

Karakter değişmezlerine örnekler: 'c', 'a', ...
İşleri basitleştirmek için, görüntülenemeyen karakterlerin girdi olarak kullanılmaması gerektiğini lütfen unutmayın. Eğer char'a dönüşüm görüntülenemiyorsa, bilgilendirici bir mesaj yazdırır.

int değişmezlerine örnekler: 0, -42, 42...

Float değişmezlere örnekler: 0.0f, -4.2f, 4.2f...
Bu sözde değişmezleri de ele almanız gerekir (bilirsiniz, bilim için): -inff, +inff ve nanf.

Çift değişmezlere örnekler: 0.0, -4.2, 4.2...

Bu sözde değişmezleri de ele almanız gerekir (bilirsiniz, eğlence için): -inf, +inf ve nan.

Sınıfınızın beklendiği gibi çalıştığını test etmek için bir program yazın.


Önce parametre olarak aktarılan değişmezin türünü tespit etmeli, string türünden gerçek türüne dönüştürmeli, ardından diğer üç veri türüne **açıkça dönüştürmelisiniz**. Son olarak, sonuçları aşağıda gösterildiği gibi görüntüleyin.

Bir dönüştürme mantıklı değilse veya taşıyorsa, kullanıcıyı tür dönüştürmenin imkansız olduğu konusunda bilgilendirmek için bir mesaj görüntüleyin. Sayısal sınırları ve özel değerleri işlemek için ihtiyacınız olan tüm başlıkları ekleyin.

```
./convert 0
char:
Görüntülenemez int:
0
float: 0.0f
double: 0.0
./convert nan
char: imkansız
int: imkansız
float: nanf
double: nan
./convert 42.0f
char: '42'
int: 42
float: 42.0f
double: 42.0
```

Bölüm V

Alıştırma 01: Serileştirme

	Egzersiz : 01
Serileştirme	
Giriş dizini : <i>ex01/</i>	
Teslim edilecek dosyalar : Makefile, *.cpp, *.h, hpp}	
Yasak fonksiyonlar : Hiçbiri	

Aşağıdaki yöntemlerle statik bir Serializer sınıfı uygulayın:

```
uintptr_t serialize(Data* ptr);
```

Bir işaretçi alır ve onu uintptr_t işaretli tamsayı türüne dönüştürür.

```
Data* deserialize(uintptr_t raw);
```

İşaretsiz bir tamsayı parametresi alır ve bunu Data'ya bir işaretçiye dönüştürür.

Sınıfınızın beklendiği gibi çalıştığını test etmek için bir program yazın.


Boş olmayan (veri üyelerine sahip olduğu anlamına gelir) bir Veri yapısı oluşturmalısınız.

Data nesnesinin adresinde serialize() işlevini kullanın ve geri dönüş değerini deserialize() işlevine aktarın. Ardından, deserialize() işlevinin geri dönüş değerinin orijinal işaretçiye eşit olduğundan emin olun.

Veri yapınıza ait dosyaları teslim etmeyi unutmayın.

Bölüm VI

Alıştırma 02: Gerçek türü tanımlama

	Alıştırma : 02
Gerçek türü tanımlayın	
Giriş dizini : <i>ex02/</i>	
Teslim edilecek dosyalar : Makefile, *.cpp, *.{h, hpp}	
Yasak işlevler : std::typeinfo	

Yalnızca genel sanal yıkıcısı olan bir **Base** sınıfı uygulayın. Base'den genel olarak miras alan üç boş **A**, **B** ve **C** sınıfı oluşturun.



Bu dört sınıfın Ortodoks Kanonik Formunda tasarlanmış olması gerekmez.

Aşağıdaki işlevleri uygulayın:

Base * generate(void);

A, B veya C'yi rastgele örneklendirir ve örneği bir Base işaretçisi olarak döndürür.

Rastgele seçim uygulaması için istediğiniz herhangi bir şeyi kullanmaktan

ç e k i n m e y i n .

void identify(Base* p);

p tarafından işaret edilen nesnenin gerçek türünü yazdırır: "A", "B" veya "C".

void tanımla(Base& p);

p tarafından işaret edilen nesnenin gerçek türünü yazdırır: "A", "B" veya "C". Bu fonksiyon içinde bir işaretçi kullanmak yasaktır.

typeinfo başlığının dahil edilmesi yasaktır.

Her şeyin beklendiği gibi çalıştığını test etmek için bir program yazın.