



C++ - Modül 01

Bellek tahsisi, üyelere işaretçiler, referanslar, switch
deyimi

Özet:

Bu doküman C++ modüllerinden Modül 01'in alıştırma larını içermektedir.

Sürüm: 10

İçindekiler

I	Giriş	2
II	Genel kurallar	3
III	Egzersiz 00: Braiiiiiinnnnzzzz	5
IV	Alıştırma 01: Daha fazla beyin!	6
V	Alıştırma 02: MERHABA BU BEYİN	7
VI	Alıştırma 03: Gereksiz şiddet	8
VII	Alıştırma 04: Sed kaybedenler içindir	10
VIII	Alıştırma 05: Harl 2.0	11
IX	Alıştırma 06: Harl filtresi	13
X	Sunum ve akran değerlendirmesi	14

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 birçok 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 oku

- 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 alıştırmamanın yönergeleri kısa görünebilir ancak örnekler, yönergelerde açıkça yazılmayan gereksinimleri gösterebilir.
- Başlamadan önce her modülü tamamen okuyun! Gerçekten okuyun.
- Odin tarafından, Thor tarafından! Beyninizi kullanın!!!




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

Egzersiz 00: BraiiiiiinnnnzzzzZ

	Egzersiz : 00
	BraiiiiiinnnnzzzzZ
	Giriş dizini : <i>ex00/</i>
	Teslim edilecek dosyalar : Makefile, main.cpp, Zombie.{h, hpp}, Zombie.cpp, newZombie.cpp, randomChump.cpp
	Yasak fonksiyonlar : Hiçbiri

İlk olarak, bir **Zombie** sınıfı oluşturun. Bir dize özel nitelik adına sahiptir. Zombi sınıfına bir void announce(void); üye işlevi ekleyin. Zombiler kendilerini aşağıdaki gibi duyururlar:

<isim>: BraiiiiiinnnnzzzzZ...

Açılı parantezleri (< ve >) yazdırmayın. Foo adlı bir zombi için mesaj şöyle olacaktır:

Foo: BraiiiiiinnnnzzzzZ...

Ardından, aşağıdaki iki işlevi uygulayın:


- **Zombi* newZombie(std::string name);**
Bir zombi yaratır, onu adlandırır ve fonksiyon kapsamı dışında kullanabilmeniz için geri döndürür.
- **void randomChump(std::string name);**
Bir zombi yaratır, ona isim verir ve zombi kendini duyurur.

Şimdi, alıştırmanın asıl amacı nedir? Hangi durumda zombileri yığına ya da heap'e ayırmanın daha iyi olacağını belirlemelisiniz.

Zombiler artık onlara ihtiyacınız olmadığında yok edilmelidir. Yok edici, hata ayıklama amacıyla zombinin adını içeren bir mesaj yazdırmalıdır.

Bölüm IV

Alıştırma 01: Daha fazla beyin!

	Egzersiz : 01
Daha çok beyin!	
Giriş dizini : <i>ex01/</i>	
Teslim edilecek dosyalar : Makefile, main.cpp, Zombie.{h, hpp}, Zombie.cpp, zombieHorde.cpp	
Yasak fonksiyonlar : Hiçbiri	

Zombi sürüsü yaratma zamanı!

Aşağıdaki işlevi uygun dosyada uygulayın:

```
Zombi*    zombieHorde( int N, std::string name );
```


Tek bir atamada N Zombi nesnesi ayrılmalıdır. Ardından, her birine parametre olarak aktarılan adı vererek zombileri başlatması gerekir. Fonksiyon ilk zombiye bir gösterici döndürür. zombieHorde() işlevinizin beklendiği gibi çalıştığından emin olmak için kendi testlerinizi uygulayın.

Her bir zombi için announce() işlevini çağırmaya çalışın.

Tüm zombileri silmeyi ve **bellek sızıntılarını** kontrol etmeyi unutmayın.

Bölüm V

Alıştırma 02: MERHABA BU BEYİN

	Alıştırma : 02
MERHABA, BEN BEYİN.	
Dönüş dizini : <i>ex02/</i>	
Teslim edilecek dosyalar : Makefile, main.cpp	
Yasak fonksiyonlar : Hiçbiri	

İçeren bir program yazın:

- "HI THIS IS BRAIN" olarak başlatılmış bir dize değişkeni.
- stringPTR: Dizeye bir işaretçi.
- stringREF: Dizeye bir referans.

Programınız yazdırmak zorunda:

- Dize değişkeninin bellek adresi.
- stringPTR tarafından tutulan bellek adresi.
- stringREF tarafından tutulan bellek adresi.


Ve sonra:

- Dize değişkeninin değeri.
- stringPTR tarafından işaret edilen değer.
- stringREF tarafından işaret edilen değer.

Hepsi bu, numara yok. Bu alıştırmanın amacı, tamamen yeni görünebilen referansların gizemini çözmektir. Bazı küçük farklılıklar olsa da, bu zaten yaptığınız bir şey için başka bir sözdizimidir: adres manipülasyonu.

Bölüm VI

Alıştırma 03: Gereksiz şiddet

	Egzersiz : 03
	Gereksiz şiddet
	Dönüş dizini : <i>ex03/</i>
	Teslim edilecek dosyalar : Makefile, main.cpp, Weapon.{h, hpp}, Weapon.cpp, HumanA.{h, hpp}, HumanA.cpp, HumanB.{h, hpp}, HumanB.cpp
	Yasak fonksiyonlar : Hiçbiri

Şu özelliklere sahip bir Silah sınıfı uygulayın:

- Bir dize olan özel bir öznitelik türü.
- type ögesine const başvuru döndüren getType() üye işlevi.
- Parametreler olarak aktarılan yeni türü kullanarak türü ayarlayan setType() üye işlevi.

Şimdi iki sınıf oluşturun: **HumanA** ve **HumanB**. Her ikisinin de bir Silahı ve bir adı var. Ayrıca attack() üye fonksiyonuna sahipler (tabii ki köşeli parantezler olmadan):

<isim> <silah türü> ile saldırır

HumanA ve HumanB, bu iki küçük ayrıntı dışında neredeyse aynıdır:

- HumanA kurucusunda Silah'ı alırken, HumanB almaz.
- HumanB'nin **her zaman** bir Silahı **olmayabilir**, oysa HumanA **her zaman** silahlı olacaktır.

Uygulamanız doğruysa, aşağıdaki kodu çalıştırdığınızda her iki test durumu için de "kaba çivili sopa" ile bir saldırı ve ardından "başka bir tür sopa" ile ikinci bir saldırı yazdırılacaktır:

```
int main()
{
    {
        Silah kulübü = Silah("kaba çivili sopa");

        HumanA bob("Bob", club);
        bob.attack();
        club.setType("başka tür bir kulüp"); bob.attack();
    }
    {
        Silah kulübü = Silah("kaba çivili sopa");

        HumanB jim("Jim");
        jim.setWeapon(club);
        jim.attack();
        club.setType("başka tür bir kulüp"); jim.attack();
    }
}

0 döndür;
```


Bellek sızıntılarını kontrol etmeyi unutmayın.



Hangi durumda Silah için bir işaretçi kullanmanın en iyisi olacağını düşünüyorsunuz? Ve Silah'a bir referans? Neden? Bu alıştırmaya başlamadan önce düşünün.

Bölüm VII

Alıştırma 04: Sed kaybedenler içindir

	Egzersiz : 04
Sed kaybedenler içindir	
Dönüş dizini : <i>ex04/</i>	
Teslim edilecek dosyalar : Makefile, main.cpp, *.cpp, *.{h, hpp}	
Yasaklı işlevler : std::string::replace	

Aşağıdaki sırada üç parametre alan bir program oluşturun: bir dosya adı ve s1 ve s2 olmak üzere iki dize.


<dosyaadı> dosyasını açacak ve içeriğini yeni bir dosyaya kopyalayacaktır
<dosyaadı>.replace, s1'in her geçtiği yeri s2 ile değiştirir.

C dosya manipülasyon fonksiyonlarını kullanmak yasaktır ve hile olarak kabul edilecektir. std::string sınıfının replace hariç tüm üye fonksiyonlarına izin verilir. Bunları akıllıca kullanın!

Elbette, beklenmedik girdileri ve hataları ele alın. Programınızın beklendiği gibi çalıştığından emin olmak için kendi testlerinizi oluşturmali ve teslim etmelisiniz.

Bölüm VIII Alıştırma

05: Harl 2.0

	Egzersiz : 05
	Harl 2.0
	Dönüş dizini : <i>ex05/</i>
	Teslim edilecek dosyalar : <i>Makefile, main.cpp, Harl.{h, hpp}, Harl.cpp</i>
	Yasak fonksiyonlar : <i>Hiçbiri</i>

Harl'ı tanıyor musun? Hepimiz tanıyoruz, değil mi? Tanımıyorsanız, Harl'ın yaptığı yorumların türlerini aşağıda bulabilirsiniz. Bunlar seviyelere göre sınıflandırılmıştır:

- **"DEBUG"** seviyesi: Hata ayıklama mesajları bağlamsal bilgiler içerir. Çoğunlukla sorun teşhisi için kullanılırlar.
Örnek: *"7XL-double-cheese-triple-pickle-special- ketchup burgerim için ekstra pastırma olmasını seviyorum. Gerçekten seviyorum!"*
- **"INFO"** seviyesi: Bu mesajlar kapsamlı bilgiler içerir. Bir üretim ortamında programın yürütülmesini izlemek için faydalıdır.
Örnek: *"Fazladan pastırma eklemenin daha fazla paraya mal olduğuna inanamıyorum. Burgerime yeterince pastırma koymamışsınız! Koymuş olsaydın, daha fazlasını istemezdim!"*
- **"UYARI"** seviyesi: Uyarı mesajları sistemde potansiyel bir sorun olduğunu gösterir. Ancak, ele alınabilir veya göz ardı edilebilir.
Örnek: *"Bence fazladan bedava pastırma yemeyi hak ediyorum. Ben yıllardır geliyorum, oysa sen geçen ay burada çalışmaya başladın."*
- **"HATA"** seviyesi: Bu mesajlar kurtarılamaz bir hata oluştuğunu gösterir. Bu genellikle manuel müdahale gerektiren kritik bir sorundur.
Örnek: *"Bu kabul edilemez! Hemen müdürle konuşmak istiyorum."*

Harl'ı otomatikleştireceksin. Her zaman aynı şeyleri söylediği için zor olmayacak. Aşağıdaki özel üye fonksiyonlara sahip bir **Harl** sınıfı oluşturmalısınız:

- void debug(void);
- void info(void);
- void uyarı(void);
- void error(void);

Harl ayrıca parametre olarak aktarılan seviyeye bağlı olarak yukarıdaki dört üye fonksiyonu çağıran bir genel üye fonksiyona sahiptir:


`geçersizcomplain(std::string level);`

Bu alıştırmanın amacı **üye fonksiyonlara işaretçi** kullanmaktır. Bu bir öneri değildir. Harl, bir if/else if/else ormanı kullanmadan şikayet etmek zorundadır. İki kez düşünmez!

Harl'ın çok şikayet ettiğini göstermek için testler oluşturun ve teslim edin. Yukarıda listelenen yorum örneklerini konu içinde kullanabilir veya kendi yorumlarınızı kullanmayı tercih edebilirsiniz.

Bölüm IX

Alıştırma 06: Harl filtresi

	Egzersiz : 06
	Harl filtre
	Dönüş dizini : <i>ex06/</i>
	Teslim edilecek dosyalar : Makefile, main.cpp, Harl.{h, hpp}, Harl.cpp
	Yasak fonksiyonlar : Hiçbiri

Bazen Harl'ın söylediği her şeye dikkat etmek istemezsiniz. Dinlemek istediğiniz günlük seviyelerine bağlı olarak Harl'ın söylediklerini filtrelemek için bir sistem uygulayın.

Parametre olarak dört seviyeden birini alan bir program oluşturun. Bu seviye ve üstündeki tüm mesajları görüntüleyecektir. Örneğin:

```
$> ./harlFilter "WARNING" [
UYARI ]
Sanırım bedavaya fazladan pastırma yemeyi hak ediyorum.
Ben yıllardır geliyorum, siz ise geçen aydan beri burada çalışmaya başladınız.

[ ERROR ]
Bu kabul edilemez, hemen müdürle konuşmak istiyorum.

$> ./harlFilter "Bugün ne kadar yorgun olduğumdan emin
değilim..." [ Muhtemelen önemsiz sorunlar hakkında şikayet
ediyor ]
```

Harl ile başa çıkmanın birkaç yolu olsa da, en etkili yöntemlerden biri onu KAPATMAKTIR.

Yürütülebilir dosyanıza harlFilter adını verin.

Bu alıştırmada switch deyimini kullanmalı ve belki de keşfetmelisiniz.



Alıştırma 06'yı yapmadan bu modülü geçebilirsiniz.

Bölüm X

Sunum ve akran değerlendirmesi

Ödevinizi her zamanki gibi Git deponuzda teslim edin. Savunma sırasında yalnızca deponuzdaki çalışmalar değerlendirilecektir. Doğru olduklarından emin olmak için klasörlerinizin ve dosyalarınızın adlarını iki kez kontrol etmekten çekinmeyin.



??????????? XXXXXXXXX = \$3\$4f1b9de5b5e60c03dcb4e8c7c7e4072c