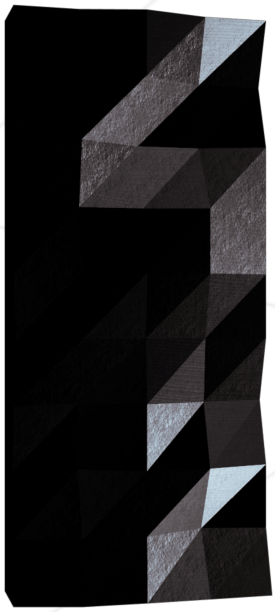

C++ - Modül 01

Memory allocation, üyeler için işaretçiler, referanslar, switch ifadesi

Özet:
Bu belge, C++ modüllerinden Modül 01 alıştırmalarını içerir.

Versiyon: 9

İçindekiler

Bölüm I

Giriş

C++, Bjarne Stroustrup tarafından C programlama dilinin bir uzantısı olarak veya "C with Classes" tarafından oluşturulan 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 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 Common Core'dan 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

Exercise 00: BraiiiiiiinnnzzzZ

	Exercise : 00
BraiiiiiiinnnzzzZ	
Turn-in directory : $ex00/$	
Files to turn in : `Makefile, main.cpp, Zombie.{h, hpp}, Zombie.cpp, newZombie.cpp, randomChump.cpp`	
Forbidden functions : `None`	

İlk olarak bir **Zombie** sınıfı implement edin. `Name` adında private bir değişkene sahiptir.

Zombi sınıfına `void announce( void );` adında bir üye fonksiyon ekleyin. Zombiler kendilerini şu şekilde duyuruyor:

```
<name>: BraiiiiiiinnnzzzZ...
```

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

```
Foo: BraiiiiiiinnnzzzZ...
```

Ardından, aşağıdaki iki işlevi uygulayın:

- `Zombie* newZombie( std::string name );`
  Bu tanım bir zombi yaratır, adlandırır ve işlev alanı dışında kullanabilmeniz için geri döndürür.
- `void randomChump( std::string name );`
  Bir zombi yaratır, onu adlandırır ve zombi kendini duyurur.

Şimdi, alıştırmanın asıl amacı nedir? Zombileri stack veya heap'e ayırmanın hangi durumda daha iyi olduğunu belirlemelisin.

Artık onlara ihtiyacınız olmadığında zombiler yok edilmelidir. Yıkıcı, hata ayıklama amacıyla zombinin adını içeren bir mesaj yazdırmalıdır.

Bölüm IV

Egzersiz 01: Moar brainz!

	Exercise : 01
Moar brainz!	
Turn-in directory : $ex01/$	
Files to turn in : `Makefile, main.cpp, Zombie.{h, hpp}, Zombie.cpp, zombieHorde.cpp`	
Forbidden functions : `None`	

Zombi sürüsü yaratma zamanı!

Uygun dosyada aşağıdaki işlevi uygulayın:

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

`N` Zombie nesnelerini tek bir tahsiste tahsis etmelidir. Ardından, her birine parametre olarak name yollayarak zombileri başlatması gerekir. İşlev, ilk zombiye bir işaretçi döndürür.

`zombieHorde()` işlevinizin beklendiği gibi çalıştığından emin olmak için kendi testlerinizi uygulayın. Zombilerin her biri için `announce()` çağırmayı deneyin.

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

Bölüm V

Egzersiz 02: HI THIS IS BRAIN

	Exercise : 02
HI THIS IS BRAIN	
Turn-in directory : $ex02/$	
Files to turn in : `Makefile, main.cpp`	
Forbidden functions : `None`	

Aşağıdakileri içeren bir program yazın:

- "HI THIS IS BRAIN" olarak başlatılan bir string değişkeni
- `stringPTR`: String için bir işaretçi.
- `stringREF`: String için bir referans

Programınızın yazdırması gerekenler:

- String değişkeninin bellek adresi.
- `stringPTR` tarafından tutulan bellek adresi.
- `stringREF` tarafından tutulan bellek adresi.

Ve daha sonra:

- String değişkeninin değeri.
- `stringPTR` tarafından işaret edilen değer.
- `stringREF` tarafından işaret edilen değer.

Hepsi bu, bir numarası yok. Bu alıştırmanın amacı, tamamen yeni gibi görünen referansların gizemini çözmektir. Bazı küçük farklılıklar olsa da, bu zaten yaptığınız bir şey için başka bir sözdizimidir: adres işleme.

Bölüm VI

Egzersiz 03: Gereksiz şiddet

	Exercise : 03
Gereksiz şiddet	
Turn-in directory : *ex*03/	
Files to turn in : `Makefile, main.cpp, Weapon.{h, hpp}, Weapon.cpp, HumanA.{h, hpp}, HumanA.cpp, HumanB.{h, hpp}, HumanB.cpp`	
Forbidden functions : `None`	

Şu özelliklere sahip bir Silah sınıfı uygulayın:

- Bir string olan gizli bir `type` değişkeni.
- `Tür` olarak bir const referans döndüren bir `getType()` üye işlevi.
- Parametre olarak iletilen değişken ile `değişken` ayarını yapan bir `setType()` üye işlevi.

Şimdi iki sınıf oluşturun: **HumanA** ve **HumanB**. Her ikisinin de bir `Weapon` ve bir `name` değişkeni var. Ayrıca (açılı parantezler olmadan) sergileyen bir `attack()` üye işlevine de sahiptirler:

```
<name> attacks with their <weapon type> (<Ahmet> <silah türü> ile saldırdı.)
```

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

- HumanA yapılandırıcısında Weapon alırken, HumanB almaz.
- HumanB'nin **her zaman bir Silahı olmayabilir** , oysa HumanA **her zaman** silahlı olacaktır.

Uygulamanız doğruysa, aşağıdaki kodu çalıştırmak, her iki test durumu için "çivili sopa" ile bir saldırı, ardından "başka bir tür sopa" ile ikinci bir saldırı yazdıracaktır:

```
int main()
{
  {
    Weapon  club = Weapon("crude spiked club");

    HumanA bob("Bob", club);
    bob.attack();
    club.setType("some other type of club");
    bob.attack();
  }
  {
    Weapon  club = Weapon("crude spiked club");

    HumanB jim("Jim");
    jim.setWeapon(club);
    jim.attack();
    club.setType("some other type of club");
    jim.attack();
  }

  return 0;
}
```

Bellek sızıntılarını kontrol etmeyi unutmayın.

Hangi durumda Silah için bir işaretçi ve silaha bir referans kullanmanın en iyisi olacağını düşünüyorsunuz? Niye? Bu alıştırmaya başlamadan önce bir düşünün.

Bölüm VII

Egzersiz 04: Üzülmek kaybedenler içindir

	Exercise : 04
Üzülmek kaybedenler içindir	
Turn-in directory : $ex04/$	
Files to turn in : `Makefile, main.cpp, *.cpp, *.{h, hpp}`	
Forbidden functions : `std::string::replace`	

Aşağıdaki sırayla üç parametre alan bir program oluşturun: bir dosya adı ve iki string, `s1` ve `s2`.

`<filename>` dosyasını açar ve içeriğini yeni bir dosyaya kopyalar

`<filename>.replace`, her `s1` ile `s2` ile.

C dosyası manipülasyon işlevlerinin kullanılması yasaktır ve kopya olarak kabul edilecektir. `replace` hariç, `std::string` sınıfının tüm üye işlevlerine izin verilir. Onları akıllıca kullanın!

Tabii ki, beklenmeyen girdileri ve hataları ele alın. Programınızın beklendiği gibi çalıştığından emin olmak için kendi testlerinizi oluşturmalı ve teslim etmelisiniz.

Bölüm VIII

Egzersiz 05: Harl 2.0

	Exercise : 05
Harl 2.0	
Turn-in directory : *ex*05/	
Files to turn in : `Makefile, main.cpp, Harl.{h, hpp}, Harl.cpp`	
Forbidden functions : `None`	

Harl'ı tanıyor musun? Hepimiz biliyoruz, değil mi? Aksi takdirde, Harl'ın yaptığı yorum türlerini aşağıda bulabilirsiniz. Onlar seviyelere göre sınıflandırılırlar:

- "**DEBUG**" düzeyi: Hata ayıklama mesajları bağlamsal bilgiler içerir. Çoğunlukla sorun teşhisi için kullanılırlar.
  Örnek: *"7XL-çift peynirli-üçlü turşu-özel-ketçaplı burgerime fazladan domuz pastırması yemeyi seviyorum. Gerçekten seviyorum!"*

- "**INFO**" düzeyi: Bu mesajlar kapsamlı bilgiler içerir. Bir üretim ortamında programın yürütülmesini izlemek için faydalıdırlar.
  Örnek: *"Ekstra domuz pastırması eklemenin daha fazla paraya mal olduğuna inanamıyorum. Burgerime yeterince pastırma koymadınız! Yapsaydınız, daha fazlasını istemezdim!"*

- "**WARNING**" düzeyi: Uyarı mesajları, sistemde olası bir sorunu belirtir.,O ele alınabilir veya yok sayılabilir.
  Örnek: *"Bence bedavaya fazladan pastırma yemeyi hak ediyorum. Ben yıllardır geliyorum, sen geçen aydan beri burada çalışmaya başladın."*

- "**ERROR**" düzeyi: Bu mesajlar, kurtarılamaz bir hatanın oluştuğunu gösterir. Bu genellikle manuel müdahale gerektiren kritik bir sorundur.
  Örnek: *"Bu kabul edilemez! Şimdi müdürle konuşmak istiyorum."*

Harl'ı otomatikleştireceksin. Hep aynı şeyleri söylediği için zor olmayacak. Aşağıdaki özel üye işlevleriyle bir **Harl** sınıfı oluşturmanız gerekir:

- `void debug( void );`
- `void info( void );`
- `void warning( void );`
- `void error( void );`

Harl ayrıca parametre olarak gönderilen düzeye bağlı olarak yukarıdaki dört üye işlevini çağıran bir genel üye işlevine sahiptir:

```
void    complain( std::string level );
```

Bu alıştırmanın amacı, **üye işlevlerine işaretçiler** kullanmaktır. Bu bir öneri değil. Harl, if/else if/else ormanını kullanmadan şikayet etmek zorundadır. İki kere düşünmez! Create and turn in tests to show that Harl complains a lot. You can use the example comments.

Bölüm IX

Egzersiz 06:Harl filtresi

	Exercise : 06
Harl filtresi	
Turn-in directory : $ex06/$	
Files to turn in : `Makefile, main.cpp, Harl.{h, hpp}, Harl.cpp`	
Forbidden functions : `None`	

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

Dört seviyeden birini parametre olarak alan bir program oluşturun. Bu seviye ve üzerindeki tüm mesajları gösterecektir. Örneğin:

```
$> ./harlFilter "WARNING"
[ WARNING ]
I think I deserve to have some extra bacon for free.
I've been coming for years whereas you started working here since last month.

[ ERROR ]
This is unacceptable, I want to speak to the manager now.

$> ./harlFilter "I am not sure how tired I am today..."
[ Probably complaining about insignificant problems ]
```

Harl ile başa çıkmanın birkaç yolu olmasına rağmen, en etkili yollardan biri onu KAPATMAKTIR.

Yürütülebilir dosyanıza `harlFilter` adını verin.

Bu alıştırmada switch ifadesini kullanmalı ve belki de keşfetmelisiniz.

Bu modülü egzersiz 06 yapmadan geçebilirsiniz.