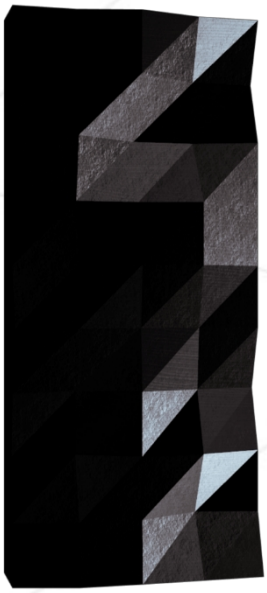

C++ - Modül 09

STL

Özet:
Bu doküman C++ modüllerinden Modül 09'un alıştırmalarını içermektedir.

Sürüm: 1.5

İçindekiler

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 çıktıda görüntülenmelidir.
- *Güle güle Norminette!* C++ modüllerinde herhangi bir kodlama stili zorunlu değildir. En sevdiğinizi 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 namespace <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. Bununla birlikte, **include korumaları** ekleyerek çift 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 alıştırmanın 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 kodlayamadığınız sürece bu sıkıcı görünebilir.

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

Bölüm III

Modüle özgü kurallar

Bu modüldeki her bir alıştırmayı gerçekleştirmek için standart konteynerlerin kullanılması zorunludur.

Bir konteyner kullanıldıktan sonra modülün geri kalanında onu kullanamazsınız.

Alıştırmaları yapmadan önce konunun tamamının okunması tavsiye edilir.

İki kap kullanılmasını gerektiren 02 numaralı egzersiz haricinde her egzersiz için en az bir kap kullanmalısınız.

Her program için kaynak dosyalarınızı -Wall, -Wextra ve -Werror bayrakları ile gerekli çıktıya derleyecek bir Makefile göndermelisiniz.

C++ kullanmanız ve Makefile dosyanızın yeniden bağlanmaması gerekir.

Makefile dosyanız en azından $(NAME), all, clean, fclean ve re kurallarını içermelidir.

Bölüm IV

Alıştırma 00: Bitcoin Borsası

	Egzersiz : 00
Bitcoin Borsası	
Giriş dizini : *ex00/*	
Teslim edilecek dosyalar : `Makefile, main.cpp, BitcoinExchange.{cpp, hpp}`	
Yasak fonksiyonlar : `Hiçbiri`	

Belirli bir tarihteki belirli bir bitcoin miktarının değerini veren bir program oluşturmanız gerekir.

Bu program, zaman içinde bitcoin fiyatını temsil edecek csv formatında bir veritabanı kullanmalıdır. Bu veritabanı bu konu ile birlikte verilmektedir.

Program, değerlendirilecek farklı fiyatları/tarihleri depolayan ikinci bir veritabanını girdi olarak alacaktır.

Programınız bu kurallara uymalıdır:

- Program adı btc.
- Programınız argüman olarak bir dosya almalıdır.
- Bu dosyadaki her satır aşağıdaki formatı kullanmalıdır: "tarih | değer".
- Geçerli bir tarih her zaman aşağıdaki formatta olacaktır: Yıl-Ay-Gün.
- Geçerli bir değer ya bir float ya da 0 ile 1000 arasında pozitif bir tamsayı olmalıdır.

`Bu alıştırmayı doğrulamak için kodunuzda en az bir konteyner kullanmalısınız. Olası hataları uygun bir hata mesajı ile ele almalısınız.`

İşte bir input.txt dosyası örneği:

```
$> head input.txt
tarih | değer
2011-01-03 | 3
2011-01-03 | 2
2011-01-03 | 1
2011-01-03 | 1.2
2011-01-09 | 1
2012-01-11 | -1
2001-42-42
2012-01-11 | 1
2012-01-11 | 2147483648
$>
```

Programınız giriş dosyanızdaki değeri kullanacaktır.

Programınız, veritabanınızda belirtilen tarihe göre döviz kuru ile çarpılan değerin sonucunu standart çıktıda göstermelidir.

Girişte kullanılan tarih DB'nizde mevcut değilse, DB'nizde bulunan en yakın tarihi kullanmanız gerekir. Üstteki tarihi değil alttaki tarihi kullanmaya dikkat edin.

Aşağıda programın kullanımına ilişkin bir örnek yer almaktadır.

```
$> ./btc
Hata: dosya açılamadı.
$> ./btc input.txt
2011-01-03 => 3 = 0,9
2011-01-03 => 2 = 0.6
2011-01-03 => 1 = 0.3
2011-01-03 => 1.2 = 0.36
2011-01-09 => 1 = 0.32
Hata: pozitif bir sayı değil. Hata:
hatalı girdi => 2001-42-42 2012-
01-11 => 1 = 7.1
Hata: çok büyük bir sayı.
$>
```

Uyarı: Bu alıştırmayı doğrulamak için kullandığınız konteyner(ler) artık bu modülün geri kalanı için kullanılamayacaktır.

Bölüm V

Alıştırma 01: Ters Lehçe Notasyonu

	Egzersiz : 01
RPN	
Giriş dizini : *ex01/*	
Teslim edilecek dosyalar : `Makefile, main.cpp, RPN.{cpp, hpp}`	
Yasak fonksiyonlar : `Hiçbiri`	

Bu kısıtlamalara sahip bir program oluşturmalısınız:

- Program adı RPN'dir.
- Programınız ters çevrilmiş bir Lehçe matematiksel ifadeyi argüman olarak almalıdır.
- Bu işlemde kullanılan ve argüman olarak aktarılan sayılar her zaman 10'dan küçük olacaktır. Hem hesaplamanın kendisi hem de sonuç bu kuralı dikkate almaz.
- Programınız bu ifadeyi işlemeli ve standart çıktıda doğru sonucu vermelidir.
- Programın yürütülmesi sırasında bir hata oluşursa, standart çıktıda bir hata mesajı görüntülenmelidir.
- Programınız bu belirteçlerle işlem yapabilmelidir: "+ - / *".

`Bu alıştırmayı doğrulamak için kodunuzda en az bir konteyner kullanmalısınız.`

Parantezleri veya ondalık sayıları yönetmenize gerek yoktur.

İşte standart bir kullanım örneği:

```
$> ./RPN "8 9 * 9 - 9 - 9 - 4 - 1 +"
42
$> ./RPN "7 7 * 7 -"
42
$> ./RPN "1 2 * 2 / 2 * 2 4 - +"
0
$> ./RPN "(1 + 1)"
Hata
$>
```

Uyarı: Önceki alıştırmada kullandığınız konteyner(ler) burada yasaklanmıştır. Bu alıştırmayı doğrulamak için kullandığınız konteyner(ler) bu modülün geri kalanında kullanılamayacaktır.

Bölüm VI

Alıştırma 02: PmergeMe

	Alıştırma : 02
PmergeMe	
Dönüş dizini : *ex02/*	
Teslim edilecek dosyalar : `Makefile, main.cpp, PmergeMe.{cpp, hpp}`	
Yasak fonksiyonlar : `Hiçbiri`	

Bu kısıtlamalara sahip bir program oluşturmalısınız:

- Programın adı PmergeMe'dir.
- Programınız argüman olarak pozitif bir tamsayı dizisi kullanabilmelidir.
- Programınız pozitif tamsayı dizisini sıralamak için birleştirme-ekleme sıralama algoritmasını kullanmalıdır.

Açıklığa kavuşturmak gerekirse, evet, Ford-Johnson algoritmasını kullanmanız gerekir.

- Programın yürütülmesi sırasında bir hata oluşursa, standart çıktıda bir hata mesajı görüntülenmelidir.

Bu alıştırmayı doğrulamak için kodunuzda en az iki farklı konteyner kullanmalısınız. Programınız en az 3000 farklı tamsayıyı işleyebilmelidir.

Algoritmanızı her konteyner için uygulamanız ve böylece genel bir fonksiyon kullanmaktan kaçınmanız şiddetle tavsiye edilir.

Standart çıktıda satır satır görüntülemeniz gereken bilgilerle ilgili bazı ek yönergeler aşağıda verilmiştir:

- İlk satırda açık bir metin ve ardından sıralanmamış pozitif tamsayı dizisini görüntülemelisiniz.
- İkinci satırda açık bir metin ve ardından sıralanmış pozitif tamsayı dizisini görüntülemelisiniz.
- Üçüncü satırda, pozitif tamsayı dizisini sıralamak için kullanılan ilk kabı belirterek algoritmanız tarafından kullanılan süreyi belirten açık bir metin görüntülemelisiniz.
- Son satırda, pozitif tamsayı dizisini sıralamak için kullanılan ikinci kabı belirterek algoritmanız tarafından kullanılan süreyi belirten açık bir metin görüntülemelisiniz.

Sıralamanızı gerçekleştirmek için kullanılan sürenin gösterilme biçimi serbesttir, ancak seçilen hassasiyet kullanılan iki kap arasındaki farkın açıkça görülmesine izin vermelidir.

İşte standart bir kullanım **örneği:**

```
$> ./PmergeMe 3 5 9 7 4
Önce: 3 5 9 7 4
Sonra:    3 4 5 7 9
Bir dizi işlemi gerçekleştirme süresi         std::[..] ile 5 öğe : 0.00031
us Bir dizi öğeyi işleme süresi   5 öğe ile std::[..] : 0.00014 us
$> ./PmergeMe `shuf -i 1-100000 -n 3000 | tr "\n" " "`
Önce: 141 79 526 321 [...]
Sonra:    79 141 321 526 [...]
std::[..] ile 3000 öğelik bir aralığı işleme süresi : 62.14389 us std::[..] ile
3000 öğelik bir aralığı işleme süresi : 69.27212 us
$> ./PmergeMe "-1" "2" Hata
$> # OSX KULLANICISI için:
$> ./PmergeMe `jot -r 3000 1 100000 | tr '\n' ' '` [...]
$>
```

Bu örnekte zamanın belirtilmesi kasıtlı olarak gariptir. Elbette, hem sıralama hem de veri yönetimi kısmı olmak üzere tüm işlemlerinizi gerçekleştirmek için kullanılan süreyi belirtmeniz gerekir.

Uyarı: Önceki alıştırmalarda kullandığınız kap(lar) burada yasaktır.

Yinelemelerle ilgili hataların yönetimi sizin takdirinize bırakılmıştır.

Bölüm VII

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.