

C++ - Modül 09

STL

Özet: Bu belge, C++ modüllerinden Modül 09'un alı ştı rmaları nı içerir.

Sürüm: 1.4

İçindekiler

| | | |
|-----|-------------------------------------|----|
| BEN | giriş | 2 |
| III | Genel kurallar | 3 |
| III | Modüle özgü kurallar | 5 |
| IV | Alıştı rma 00: Bitcoin Borsası | 6 |
| V | Alıştı rma 01: Ters Lehçe Notasyonu | 8 |
| VI | Alıştı rma 02: PmergeMe | 10 |
| VII | Sunum ve meslektaş değerlendirmesi | 13 |

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 Nesne Yönelimli Programlama ile tanı ştı rmaktır .
Bu, C++ yolculuğunuzun başlangı ç noktası olacaktır r. OOP öğrenmek için birçok dil önerilir.
Eski dostunuz C'den türetildiği için C++'ı seçmeye karar verdik.
Bu karmaşı k bir dil olduğundan ve işleri basit tutmak için kodunuz C++98 standardı na uygun olacaktır r.

Modern C++'ı n pek çok 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 işaretleri ile 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 özniteliklerinizi gerektiği gibi adlandırın. yardımcı notlar.
- Sı nı f isimlerini UpperCamelCase formatı nda yazın . Sı nı f kodu içeren dosyalar her zaman sı nı f adı na göre adlandırılmalıdır. Örneğin:
ClassName.hpp/ClassName.h, ClassName.cpp veya ClassName.hpp. Ardı ndan, bir tuğla duvarı temsil eden "BrickWall" sı nı fı nı tanı mı nı içeren bir başlı k dosyanı z varsa, adı BrickWall.hpp olacaktır .
- Aksi belirtilmedikçe, her çıktı ş mesajı yeni bir satı rla sonlandırılmalıdır. karakter ve standart çıktı ya görüntülenir.
- Elveda Norminette! C++ modüllerinde kodlama stili uygulanmaz. En sevdiğinizi takip edebilirsiniz. Ancak, akran değerlendiricilerinizin anlayamadığı bir kodun not veremediği bir kod olduğunu unutmayın. Temiz ve okunabilir bir kod yazmak için elinizden geleni yapın .

İzin Verildi/Yasaklandı

Artı k C'de kodlama yapmı yorsunuz. C++ zamanı ! Öyleyse:

- Standart kitaplı ktan hemen hemen her şeyi kullanmanı za izin verilir. Bu nedenle, zaten bildiklerinize bağlı kalmak yerine, alışkı n olduğunuz C işlevlerinin mümkün olduğunca C++-ish sürümlerini kullanmak akı llı ca olacaktır .
- Ancak, başka bir harici kitaplı k kullanamazsı nı z. Bu, C++11 (ve türetilmiş formlar) ve Boost kitaplı kları nı n yasak olduğu anlamı na gelir. Aşağı daki işlevler de yasaktır: *printf(), *alloc() ve free(). Bunları kullanı rsanı z notunuz 0 olur ve o kadar.

- Aksi açı kça belirtilmediği sürece <ns_name> ve arkadaş anahtar kelimeleri yasaktı r. Aksi takdirde notunuz -42 olacaktır r.
- STL'yi yalnızca Modül 08 ve 09'da kullanmanı za izin verilir. Bunun anlamı : o zamana kadar Konteyner (vektör/liste/harita/vb.) ve Algoritma (<algorithm> başlı ğı nı içermesini gerektiren herhangi bir şey) olmaması . Aksi takdirde notunuz -42 olacaktır r.

Birkaç tasarım gereksinimi

- Bellek sı zı ntı sı C++'da da meydana gelir. Bellek ayı rdı ğı nı zda (yeni anahtar kelime), bellek sı zı ntı ları ndan kaçınılmalı sı nı z.
- Modül 02'den Modül 09'a kadar sı nı fları nı z Ortodoks olarak tasarlanmalı dı r. Kanonik Form, aksi açı kça belirtilmedikçe.
- Bir başlı k dosyası na konulan herhangi bir işlev uygulaması (işlev şablonları hariç), alı ştı rma için 0 anlamı na gelir.
- Başlı kları nı zı n her birini diğerlerinden bağı msı z olarak kullanabilmelisiniz. Bu nedenle, ihtiyaç duydukları tüm bağı mlı lı kları içermeleri gerekir. Ancak, içerme korumaları ekleyerek çift içerme probleminden kaçınılmalı sı nı z. Aksi takdirde notunuz 0 olacaktır r.

beni oku

- Gerekirse (örn. kodunuzu bölmek için) bazı ek dosyalar ekleyebilirsiniz. Bu ödevler bir program tarafı ndan doğrulanmadı ğı ndan, zorunlu dosyaları teslim ettiğiniz sürece bunu yapmaktan çekinmeyin.
- Bazen, bir alı ştı rmanı n yönergeleri kı sa görünebilir, ancak örnekler şunları gösterebilir: talimatlarda açı kça yazı lmayan gereksinimler.
- Başlamadan önce her modülü baştan sona okuyun! Gerçekten, yap.
- Odin adı na, Thor adı na! Beynini kullan!!!



Çok sayı da sı nı f uygulamaları z gerekecek. Bu sı kıcı görünebilir, favori metin düzenleyicinizi yazamazsanı z.



Alı ştı rmaları tamamlamanı z için size belirli bir miktar özgürlük verilir. Ancak zorunlu kurallara uyun ve tembel olmayı n. Yapabilirdin birçok yararlı bilgiyi kaçırır n! hakkı nda okumaktan çekinmeyin teorik kavramlar.

Bölüm III

Modüle özgü kurallar

Bu modüldeki her alıştırmayı gerçekleştirmek için standart kapları kullanılması zorunludur.

Bir konteyner bir kez kullanıldıktan sonra, onu modülün geri kalanı için kullanamazsınız.



İşlemi yapmadan önce konunun tamamını okumanızı tavsiye edilir.
egzersizler.



Her egzersiz için en az bir kap kullanılmalıdır.
iki kap kullanımı gerektiren egzersiz 02 istisnadır.

Kaynak dosyaları -Wall, -Wextra ve -Werror bayrakları ile gerekli çıktıya derleyecek her program için bir Makefile göndermelisiniz.

C++ kullanılmalıdır ve Makefile'ı yeniden bağlanmamalıdır.

Makefile dosyası en azından \$(NAME), all, clean, fclean ve re kuralları içermelidir.

Bölüm IV

Alıştırma 00: Bitcoin Borsası

| | |
|---|---------------|
|  | Egzersiz : 00 |
| Bitcoin Borsası | |
| Teslim dizini : ex00/ Teslim | |
| edilecek dosyalar : Makefile, main.cpp, BitcoinExchange.{cpp, hpp} | |
| Yasaklanan işlevler: Yok | |

Belirli bir tarihte belirli miktarda bitcoin değerini veren bir program oluşturmalısınız.

Bu program, bitcoin fiyatını temsil edecek csv formatında bir veritabanı kullanmalıdır. Bu veri tabanı bu konu ile birlikte sağlanmaktadır.

Program, değerlendirmek üzere farklı fiyatları /tarihleri saklayan ikinci bir veri tabanını girdi olarak alacaktır.

Programınızı şu kurallara uymalısınız:

- Programın adı btc'dir.
- Programınızı argüman olarak bir dosya almalıdır.
- Bu dosyadaki her satır şu biçimi kullanmalıdır: "tarih | değer".
- Geçerli bir tarih her zaman şu formatta olacaktır: Yı l-Ay-Gün.
- Geçerli bir değer, kayan nokta veya 0 ile 1000 arası nda pozitif bir tam sayı olmalıdır.



Bu alıştırmayı doğrulamak için kodunuzda en az bir kapsayıcı kullanmalısınız. Olası hataları uygun bir şekilde ele almalısınız. hata mesajı .

İşte bir input.txt dosyası örneği:

```
$> kafa girişi.txt tarihi |  
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 $>
```

Programı nı z giriş dosyası ndaki değeri kullanacaktır.

Programı nı z çarpı lan değerin sonucunu standart çıktı da göstermelidir.
veri tabanı nı zda belirtilen tarihe göre döviz kuruna göre.



Girdide kullanı lan tarih DB'nizde yoksa, DB'nizde bulunan en yakı n tarihi kullanmalı sı nı z.
Üst tarihi değil alt tarihi kullanmaya dikkat edin.

Aşağı da programı n kullanı mı na bir örnek verilmiştir.


```
$> ./btc  
Hatası : dosya açıl amadı . $> ./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 giriş => 2001-42-42 2012-01-11  
=> 1 = 7,1  
Hata: çok büyük bir sayı . $>
```



Uyarı : Bu alışı rmayı doğrulamak için kullandığı nı z kap(lar) artı k bu modülün geri kalanı nda kullanı lamayacaktır.

Bölüm V

Alıştırma 01: Ters Lehçe Notasyonu

| | |
|---|--|
|  | Egzersiz : 01 |
| | RPN |
| | Teslim dizini : ex01/ Teslim |
| | edilecek dosyalar : Makefile, main.cpp, RPN.{cpp, hpp} |
| | Yasaklanan işlevler: Yok |

Bu kısıtlamalara sahip bir program oluşturmalısınız:

- Programın adı RPN'dir.
- Programınız, ters çevrilmiş bir Lehçe matematiksel ifadeyi argüman olarak almalıdır.
- Bu işlemde kullanılan ve argüman olarak iletilen sayılar her zaman 10'dan küçük olacaktır. Hesaplamanın kendisi ve sonuç da bu kurala dikkate almaz.
- Programınız bu ifadeyi işlemeli ve ekranda doğru sonucu vermelidir.
- Programın yürütülmesi sırasında bir hata oluşursa, bir hata mesajı verilmelidir.
- Programınız şu belirteçlerle işlemleri gerçekleştirebilmelidir: "+ - / *".



Bunu doğrulamak için kodunuzda en az bir kapsayıcı kullanmalısınız.
egzersiz yapmak.



Parantezleri veya ondalık sayıları yönetmenize gerek yoktur.

İşte standart kullanıma bir örnek:

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

Hata

```
$>
```



Uyarı : Bir önceki alıştırmada kullandığınız bazı kap(lar) burada yasaklanmıştır. Bu alıştırmayı doğrulamak için kullandığınız bazı kapsayıcı(lar) bu modülün geri kalanı için kullanılamaz.

Bölüm VI

Alıştırma 02: PmergeMe

| | |
|---|---|
|  | Egzersiz : 02 |
| | PmergeMe |
| | Teslim dizini : ex02/ Teslim |
| | edilecek dosyalar : Makefile, main.cpp, PmergeMe.{cpp, hpp} |
| | Yasaklanan işlevler: Yok |

Bu kı s tlamalara sahip bir program oluşturmalı s nı z:

- Programı n adı PmergeMe'dir.
- Programı nı z pozitif bir tamsayı dizisini bağı msı z değışken olarak kullanabilmelidir.
- Programı nı z, pozitif tamsayı yı sı ralamak için birleştirme-ekleme sı ralama algoritması nı kullanmalı dı r. sekans.



Açı klı ğa kavuşturmak için, evet, Ford-Johnson algoritması nı kullanmanı z gerekiyor.

- Programı n yürütülmesi sı rası nda bir hata oluşursa, standart çı ktı da bir hata mesajı görüntülenmelidir.



Bu alıştırı rmayı doğrulamak için kodunuzda en az iki farklı kap kullanmalı s nı z. Programı nı z en az 3000 farklı tamsayı işleyebilmelidir.



Algoritmanızı her kapsayıcı için uygulamayı ve böylece genel bir işlev kullanmaktan kaçınmanızı şiddetle tavsiye edilir.

Standart çıktı da satır satır görüntülemeniz gereken bilgilerle ilgili bazı ek yönergeler şunlardır:

- İlk satırda, açık bir metni ve ardından sıralanmamış olumlu metni görüntülemeniz gerekir. tamsayı dizisi.
- İkinci satırda, açık bir metni ve ardından sıralanmış olumlu metni görüntülemeniz gerekir. tamsayı dizisi.
- Üçüncü satırda, pozitif tamsayıyı sıralamak için kullanılan ilk kabı belirterek, algoritmanızı tarafımdan kullanılan zamanı gösteren açık bir metin göstermelisiniz. sekans.
- Son satırda, pozitif tamsayı dizisini sıralamak için kullanılan ikinci kapsayıcıyı belirterek, algoritmanızı tarafımdan kullanılan zamanı gösteren açık bir metin göstermelisiniz.



Sıralamanızı gerçekleştirmek için kullanılan zamanın görüntülenme formatı ücretsizdir ancak seçilen hassasiyet, kullanılan iki kap arasındaki fark.

İşte standart kullanıma bir örnek :

```
$> ./PmergeMe 3 5 9 7 4 Önce: 3
5 9 7 4 Sonra: 3 4 5 7 9
std::[...] ile 5 elemanlı k
bir aralı ğı işleme zamanı 0 -n 3000 | tr "\n" " " Önce: 141 79 526 321 [...]

Sonra: 79 141 321 526 [...]
3000 öğelik bir diziyi std::[...] ile işleme süresi: 62.14389 us 3000 öğelik bir diziyi std::[...] ile 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 kısmı hem de veri yönetimi kısmı olmak üzere tüm işlemlerinizi gerçekleştirmek için kullanılan süreyi belirtmeniz gerekir.



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



Yinelemelerle ilgili hataların yönetimi size bırakılmıştır. sağduyu.

Bölüm VII

Başvuru ve akran değerlendirmesi

Git deponuzdaki ödevinizi her zamanki gibi teslim edin. Savunma sırası anda yalnızca deponuzdaki çalışma değerlendirilecektir. Doğru oldukları ndan emin olmak için klasörlerinizin ve dosyaları nı zı n adları nı iki kez kontrol etmekten çekinmeyin.