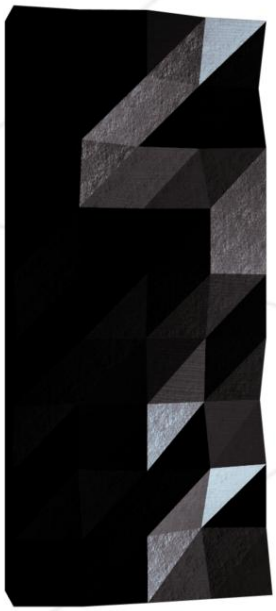

C++ - Módulo 09

STL

Resumo:
Este documento contém os exercícios do Módulo 09 dos módulos C++.

Versão: 1.5

Conteúdo

EU	**Introdução**	**2**
II	**Regras gerais**	**3**
III	**Regras específicas do módulo**	**5**
IV Exercício 00: Troca de Bitcoin		**6**
V Exercício 01: Notação Polonesa Reversa		**8**
VI Exercício 02: PmergeMe		**10**
VII Submissão e avaliação por pares		**13**

Capítulo I

Introdução

C++ é uma linguagem de programação de uso geral criada por Bjarne Stroustrup como uma extensão da linguagem de programação C, ou "C com Classes" (fonte: Wikipedia).

O objetivo desses módulos é apresentar a **Programação Orientada a Objetos.** Este será o ponto de partida de sua jornada em C++. Muitas linguagens são recomendadas para aprender OOP. Decidimos escolher C++ porque ele é derivado do seu velho amigo C. Por se tratar de uma linguagem complexa e para manter as coisas simples, seu código estará em conformidade com o padrão C++98.

Estamos cientes de que o C++ moderno é muito diferente em muitos aspectos. Portanto, se você deseja se tornar um desenvolvedor C++ proficiente, cabe a você ir além após o 42 Common Core!

Capítulo II

Regras gerais

Compilando

- Compile seu código com c++ e os sinalizadores -Wall -Wextra -Werror
- Seu código ainda deverá ser compilado se você adicionar o sinalizador -std=c++98

Convenções de formatação e nomenclatura

- Os diretórios dos exercícios serão nomeados desta forma: ex00, ex01, ... , ex
- Nomeie seus arquivos, classes, funções, funções de membro e atributos conforme exigido em As diretrizes.
- Escreva nomes de classes no formato **UpperCamelCase .** Arquivos contendo código de classe serão sempre ser nomeado de acordo com o nome da classe. Por exemplo: ClassName.hpp/ClassName.h, ClassName.cpp ou ClassName.tpp. Então, se você tiver um arquivo de cabeçalho contendo a definição de uma classe "BrickWall" que representa uma parede de tijolos, seu nome será BrickWall.hpp.
- A menos que especificado de outra forma, todas as mensagens de saída devem ser finalizadas com uma nova linha caractere e exibido na saída padrão.
- *Adeus Norminette!* Nenhum estilo de codificação é imposto nos módulos C++. Você pode seguir o seu favorito. Mas tenha em mente que um código que seus pares avaliadores não conseguem entender é um código que eles não podem avaliar. Faça o seu melhor para escrever um código limpo e legível.

Permitido/Proibido

Você não está mais codificando em C. É hora de C++! Portanto:

- Você tem permissão para usar quase tudo da biblioteca padrão. Assim, em vez de se ater ao que você já sabe, seria inteligente usar o máximo possível as versões C++ das funções C com as quais você está acostumado.
- Entretanto, você não pode usar nenhuma outra biblioteca externa. Isso significa que C++ 11 (e formas derivadas) e bibliotecas Boost são proibidas. As seguintes funções também são proibidas: *printf(), *alloc() e free(). Se você usá-los, sua nota será 0 e pronto.

- Observe que, salvo indicação explícita em contrário, o namespace using <ns_name> e palavras-chave de amigos são proibidas. Caso contrário, sua nota será -42.
- **É permitido utilizar o STL somente nos Módulos 08 e 09.** Isso significa: nenhum **contêiner** (vetor/lista/mapa/e assim por diante) e nenhum **algoritmo** (qualquer coisa que exija a inclusão do cabeçalho <algorithm>) até então. Caso contrário, sua nota será -42.

Alguns requisitos de design

- O vazamento de memória também ocorre em C++. Quando você aloca memória (usando o novo palavra-chave), você deve evitar **vazamentos de memória.**
- Do Módulo 02 ao Módulo 09, suas aulas devem ser planejadas no estilo **Ortodoxo Forma Canônica, exceto quando explicitamente indicado de outra forma.**
- Qualquer implementação de função colocada em um arquivo de cabeçalho (exceto modelos de função) significa 0 para o exercício.
- Você deve ser capaz de usar cada um dos seus cabeçalhos independentemente dos outros. Assim, eles devem incluir todas as dependências de que necessitam. No entanto, você deve evitar o problema da inclusão dupla adicionando **guardas de inclusão.** Caso contrário, sua nota será 0.

Leia-me

- Você pode adicionar alguns arquivos adicionais se precisar (ou seja, para dividir seu código). Como essas atribuições não são verificadas por um programa, fique à vontade para fazê-lo, desde que entregue os arquivos obrigatórios.
- Às vezes, as orientações de um exercício parecem curtas, mas os exemplos podem mostrar requisitos que não estão explicitamente escritos nas instruções.
- Leia cada módulo completamente antes de começar! Realmente, faça isso.
- Por Odin, por Thor! Use seu cérebro!!!

Você terá que implementar muitas classes. Isso pode parecer tedioso, a menos que você consiga criar um script em seu editor de texto favorito.

Você tem uma certa liberdade para completar os exercícios. Porém, siga as regras obrigatórias e não seja preguiçoso. Você poderia perca muitas informações úteis! Não hesite em ler sobre conceitos teóricos.

Capítulo III

Regras específicas do módulo

É obrigatório utilizar os containers padrão para realizar cada exercício deste módulo.

Depois que um contêiner for usado, você não poderá usá-lo no restante do módulo.

É aconselhável ler o assunto na íntegra antes de fazer o exercícios.

Você deve usar pelo menos um recipiente para cada exercício com o exceção do exercício 02 que exige a utilização de dois containers.

Você deve enviar um Makefile para cada programa que irá compilar seus arquivos de origem para a saída necessária com os sinalizadores -Wall, -Wextra e -Werror.

Você deve usar c++ e seu Makefile não deve ser vinculado novamente.

Seu Makefile deve conter pelo menos as regras $(NAME), all, clean, fclean e re.

Capítulo IV

Exercício 00: Troca de Bitcoin

	Exercício: 00
Troca de Bitcoins	
Diretório de entrega: ex00/	
Arquivos para entrega: Makefile, main.cpp, BitcoinExchange.{cpp, hpp}	
Funções proibidas: Nenhuma	

Você tem que criar um programa que produza o valor de uma certa quantidade de bitcoin em uma determinada data.

Este programa deve utilizar um banco de dados em formato csv que representará o preço do bitcoin ao longo do tempo. Este banco de dados é fornecido com este assunto.

O programa terá como entrada uma segunda base de dados, armazenando os diferentes preços/datas a avaliar.

Seu programa deve respeitar estas regras:

- O nome do programa é btc.
- Seu programa deve receber um arquivo como argumento.
- Cada linha deste arquivo deve utilizar o seguinte formato: "data | valor".
- Uma data válida estará sempre no seguinte formato: Ano-Mês-Dia.
- Um valor válido deve ser um número flutuante ou um número inteiro positivo, entre 0 e 1000.

Você deve usar pelo menos um contêiner em seu código para validar este exercício. Você deve lidar com possíveis erros com uma abordagem apropriada mensagem de erro.

Aqui está um exemplo de um arquivo input.txt:

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

Seu programa usará o valor em seu arquivo de entrada.

Seu programa deverá exibir na saída padrão o resultado do valor multiplicado pela taxa de câmbio de acordo com a data indicada em seu banco de dados.

Se a data usada na entrada não existir no seu banco de dados, você deverá usar a data mais próxima contida no seu banco de dados. Tenha cuidado ao usar a data inferior e não a superior.

A seguir está um exemplo de uso do programa.

```
$> ./btc Erro:
não foi possível abrir o arquivo. $> ./btc
input.txt 03/01/2011 => 3 =
0,9 03/01/2011 => 2 = 0,6

03/01/2011 => 1 = 0,3 03/01/2011
=> 1,2 = 0,36
09/01/2011 => 1 = 0,32
Erro: não é um número positivo.
Erro: entrada incorreta => 2001-42-42 2012-01-11
=> 1 = 7,1
Erro: um número muito grande. $>
```

Aviso: os contêineres que você usa para validar este exercício não poderão mais ser usados no restante deste módulo.

Capítulo V

Exercício 01: Notação Polonesa Reversa

	Exercício: 01
RPN	
Diretório de entrega: ex01/	
Arquivos para entrega: Makefile, main.cpp, RPN.{cpp, hpp}	
Funções proibidas: Nenhuma	

Você deve criar um programa com estas restrições:

- O nome do programa é RPN.
- Seu programa deve usar uma expressão matemática polonesa invertida como argumento mento.
- Os números utilizados nesta operação e passados como argumentos serão sempre menores que 10. O cálculo em si mas também o resultado não levam em conta esta regra.
- Seu programa deve processar esta expressão e gerar o resultado correto no saída padrão.
- Caso ocorra algum erro durante a execução do programa uma mensagem de erro deverá ser exibido na saída padrão.
- Seu programa deve ser capaz de lidar com operações com estes tokens: "+ - / *".

Você deve usar pelo menos um contêiner em seu código para validar isso exercício.

Você não precisa gerenciar colchetes ou números decimais.

Aqui está um exemplo de uso padrão:

```
$> ./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)"


Erro $>
```

Aviso: Os contêineres que você usou no exercício anterior são proibidos aqui. O(s) contêiner(es) que você usou para validar este exercício
não será utilizável para o restante deste módulo.

Capítulo VI

Exercício 02: PmergeMe

	Exercício: 02
PmergeMe	
Diretório de entrega: ex02/	
Arquivos para entrega: Makefile, main.cpp, PmergeMe.{cpp, hpp}	
Funções proibidas: Nenhuma	

Você deve criar um programa com estas restrições:

- O nome do programa é PmergeMe.
- Seu programa deve ser capaz de usar uma sequência inteira positiva como argumento.
- Seu programa deve usar o algoritmo de classificação por mesclagem e inserção para classificar o número inteiro positivo seqüência.

Para esclarecer, sim, você precisa usar o algoritmo Ford-Johnson.

- Se ocorrer um erro durante a execução do programa, uma mensagem de erro deverá ser exibida na saída padrão.

Você deve usar pelo menos dois contêineres diferentes em seu código para validar este exercício. Seu programa deve ser capaz de lidar com pelo menos 3.000 números inteiros diferentes.

É altamente recomendável implementar seu algoritmo para cada contêiner e, assim, evitar o uso de uma função genérica.

Aqui estão algumas diretrizes adicionais sobre as informações que você deve exibir linha por linha na saída padrão:

- Na primeira linha você deve exibir um texto explícito seguido do positivo não classificado sequência inteira.
- Na segunda linha você deve exibir um texto explícito seguido do positivo ordenado sequência inteira.
- Na terceira linha você deve exibir um texto explícito indicando o tempo utilizado pelo seu algoritmo especificando o primeiro container utilizado para ordenar o inteiro positivo seqüência.
- Na última linha você deve exibir um texto explícito indicando o tempo utilizado pelo seu algoritmo, especificando o segundo contêiner utilizado para ordenar a sequência de inteiros positivos.

O formato de visualização do tempo utilizado para realizar a sua triagem é gratuito, mas a precisão escolhida deve permitir ver claramente o diferença entre os dois recipientes usados.

Aqui está um **exemplo** de uso padrão:

```
$> ./PmergeMe 3 5 9 7 4 Antes: 3
5 9 7 4 Depois: 3 4 5 7 9
Tempo para processar um
intervalo de 5 elementos com std::[..] : 0,00031 us Tempo para processar um intervalo de 5 elementos
com std::[..] : 0.00014 us $> ./PmergeMe `shuf -i 1-100000 -n 3000 | tr "\n" " "` Antes: 141 79 526 321
[...]

Depois: 79 141 321 526 [...]
Tempo para processar um intervalo de 3.000 elementos com std::[..] : 62.14389 us Tempo para
processar um intervalo de 3.000 elementos com std::[..] : 69.27212 us $> ./PmergeMe "-1" "2"

Erro
$> # Para USUÁRIO OSX:
$> ./PmergeMe `jot -r 3000 1 100000 | tr '\n' ' '` [...] $>
```

A indicação da hora é deliberadamente estranha neste exemplo.
Claro que você deve indicar o tempo utilizado para realizar todas as suas operações, tanto a parte de classificação quanto a parte de gerenciamento de dados.

Aviso: Os contêineres que você usou nos exercícios anteriores são
proibido aqui.

A gestão de erros relacionados com duplicados é deixada ao seu critério
critério.

Capítulo VII

Envio e avaliação por pares

Entregue sua tarefa em seu repositório Git normalmente. Somente o trabalho dentro do seu repositório será avaliado durante a defesa. Não hesite em verificar os nomes de suas pastas e arquivos para garantir que estão corretos.