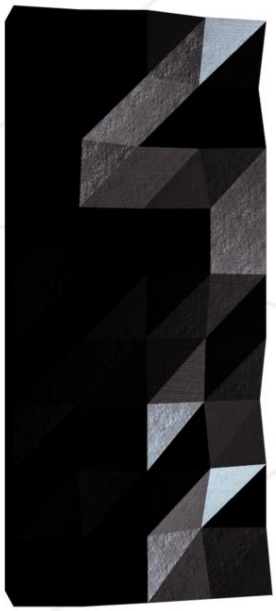

C++ - Módulo 07

Modelos C++

Resumo:
Este documento contém os exercícios do Módulo 07 dos módulos C++.

Versão: 6

Conteúdo

EU	**Introdução**	**2**
II	**Regras gerais**	**3**
III	**Exercício 00: Comece com algumas funções**	**5**
IV	**Exercício 01: Iter**	**7**
V	**Exercício 02: Matriz**	**8**

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

Exercício 00: Comece com algumas funções

	Exercício: 00
Comece com algumas funções	
Diretório de entrega: ex00/	
Arquivos para entrega: Makefile, main.cpp, qualquer que seja.{h, hpp}	
Funções proibidas: Nenhuma	

Implemente os seguintes modelos de função:

- swap: Troca os valores de dois argumentos dados. Não retorna nada.
- min: Compara os dois valores passados em seus argumentos e retorna o menor um. Se os dois forem iguais, ele retornará o segundo.
- max: Compara os dois valores passados em seus argumentos e retorna o maior. Se os dois forem iguais, ele retornará o segundo.

Essas funções podem ser chamadas com qualquer tipo de argumento. O único requisito é que os dois argumentos tenham o mesmo tipo e suportem todos os operadores de comparação.

Os modelos devem ser definidos nos arquivos de cabeçalho.

Executando o seguinte código:

```
interno principal( vazio ) {

    intuma = 2;
    int b = 3;

    ::trocar(a,b);
    std::cout << "a = << a << ", b = << b << std::endl;
    std::cout << "min( a, b ) = << ::min( a, b ) << std::endl;
    std::cout << "max( a, b ) = << ::max( a, b ) << std::endl;

    std::string c = "chaine1";
    std::string d = "chaine2";

    ::trocar(c, d);
    std::cout << "c = << c << ", d = << d << std::endl;
    std::cout << "min( c, d ) = << ::min( c, d ) << std::endl;
    std::cout << "max( c, d ) = << ::max( c, d ) << std::endl;

    retornar 0;
}
```

Deve produzir:

```
uma = 3, b = 2
min(a, b) = 2
máximo (a, b) = 3
c = cadeia2, d = cadeia1
min(c, d) = cadeia1
max(c, d) = cadeia2
```

Capítulo IV

Exercício 01: Iter

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

Implemente um iter de modelo de função que receba 3 parâmetros e não retorne nada.

• O primeiro parâmetro é o endereço de um array.

• O segundo é o comprimento do array.

• A terceira é uma função que será chamada em todos os elementos do array.

Entregue um arquivo main.cpp que contém seus testes. Forneça código suficiente para gerar um executável de teste.

Seu modelo de função iter deve funcionar com qualquer tipo de array. O terceiro parâmetro pode ser um modelo de função instanciado.

Capítulo V

Exercício 02: Matriz

	Exercício: 02
Variedade	
Diretório de entrega: ex02/	
Arquivos para entrega: Makefile, main.cpp, Array.{h, hpp} e arquivo opcional: Array.tpp Funções proibidas:	
Nenhuma	

Desenvolva um modelo de classe **Array** que contenha elementos do tipo T e que implemente o seguinte comportamento e funções:

- Construção sem parâmetro: Cria um array vazio.
- Construção com um unsigned int n como parâmetro: Cria um array de n elementos inicializado por padrão.
  Dica: Tente compilar int * a = new int(); em seguida, exiba *a.
- Construção por cópia e operador de atribuição. Em ambos os casos, a modificação do array original ou sua cópia após a cópia não deve afetar o outro array.
- Você DEVE usar o operador new[] para alocar memória. A alocação preventiva (localização antecipada de memória) é proibida. Seu programa nunca deve acessar memória não alocada.
- Os elementos podem ser acessados através do operador subscrito: [ ].
- Ao acessar um elemento com o operador [ ], se seu índice estiver fora dos limites, uma std::exception será lançada.
- Uma função membro size() que retorna o número de elementos no array. Esta função membro não aceita parâmetros e não deve modificar a instância atual.

Como de costume, certifique-se de que tudo funciona conforme o esperado e entregue um arquivo main.cpp que contém seus testes.