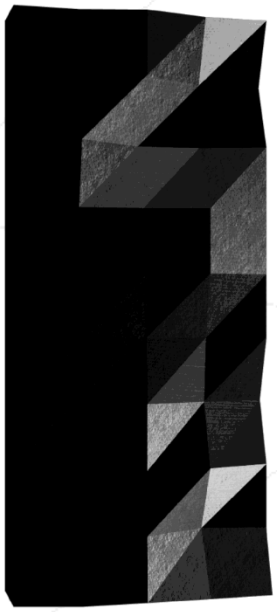

# C++ - Módulo 06

## C++ casts

*Summary:*

*Este documento contém os exercícios eletrônicos do Módulo 06 do Co módulos.*

*Versão: 5.1*

# Contents

# Capítulo III
# Introdução

*CA é uma linguagem de programação de uso geral criada por B. Jarne Stroustrup como uma ex-alternativa à linguagem de programação.*
/enston *da p r o g r a m a ç ã o C, ou "C* mttfi *Classes" (fonte.* Wtk(ped(a .

O objetivo desses módulos é apresentá-lo à programação orientada a objetos. Esse será o ponto de partida de sua jornada em C++. Muitas linguagens são recomendadas para aprender OOP. Decidimos escolher o C++, pois ele é derivado do seu velho amigo C. Como essa é uma linguagem complexa, e para manter as coisas simples, seu código obedecerá ao padrão C1+98.

Sabemos que o C++ moderno é muito diferente em vários aspectos. Portanto, se você quiser se tornar um desenvolvedor C++ proficiente, cabe a você ir além do 42 Common Core!

# Capítulo II
# Regras gerais

## Compilação

- Compile seu código com c++ e os sinalizadores -Tal l -text ra-terror
- Seu código ainda deverá ser compilado se você adicionar o fiag -

std=c++98 Convenções de formatação e nomenclatura

- Os diretórios de exercícios serão nomeados da seguinte forma: ex00 , ex01 , , exn
- Nomeie seus arquivos, classes, funções, funções-membro e atributos conforme exigido nas diretrizes.
- Escreva os nomes das classes no formato UpperCamelCase. Os arquivos que contêm código de classe sempre serão nomeados de acordo com o nome da classe. Por exemplo:
  ClassNane . hpp /ClassNane . h, ClassNane . cpp ou ClassNane . App. Então, se você tiver um arquivo de cabeçalho que contenha a definição de uma classe "BrickWall" que representa uma parede de tijolos, seu nome será Bri ckWal l . hpp.
- A menos que especificado de outra forma, todas as mensagens de saída devem ser encerradas com um caractere de nova linha e exibidas na saída padrão.
- *Adeus Norminelle!* Nenhum estilo de codificação é imposto nos módulos CH. Você pode seguir seu estilo favorito. Mas lembre-se de que um código que seus colegas avaliadores não conseguem entender é um código que eles não podem avaliar. Faça o possível para escrever um código limpo e legível.

## Permitido/Proibido

Você não está mais codificando em C. É hora de usar o C1+! Portanto:

- Você tem permissão para usar quase tudo da biblioteca padrão. Portanto, em vez de se ater ao que já sabe, seria inteligente usar o máximo possível as versões em C++ das funções em C com as quais está acostumado.
- No entanto, você não pode usar nenhuma outra biblioteca externa. Isso significa que as bibliotecas C+ +11 (e formas derivadas) e Boost são proibidas. As seguintes funções também são proibidas: *pr ink I () , *al 1 o c () e I ree ( ) . Se você as usar, sua nota será 0 e pronto.

- Observe que, a menos que explicitamente declarado de outra forma, o uso do namespace 4ns_nane e das palavras-chave friend é proibido. Caso contrário, sua nota será -42.
- Você tem permissão para usar o STL somente nos Módulos 08 e 09. Isso significa: nada de contêineres (vetor/lista/mapa/e assim por diante) e nada de algoritmos (qualquer coisa que exija a inclusão do cabeçalho algor it hut) até lá. Caso contrário, sua nota será -42.

Alguns requisitos de design

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

Leia-me

- Você pode acrescentar alguns arquivos adicionais, se necessário (ou seja, para dividir seu código). Como essas tarefas não são verificadas por um programa, sinta-se à vontade para fazer isso, desde que entregue os arquivos obrigatórios.
- Às vezes, as diretrizes 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! De fato, faça isso.
- Por Odin, por Thor! Use seu cérebro!!!

Você terá de implementar muitas classes. Isso pode parecer entediante, a menos que você saiba programar seu editor de texto favorito.

Você tem certa liberdade para concluir os exercícios. Entretanto, siga as regras obrigatórias e não seja preguiçoso. Você perderia muitas informações úteis! Não hesite em ler sobre conceitos teóricos.

# Capítulo IV III
# Regra adicional

A regra a seguir se aplica a todo o módulo e não é opcional.

Para cada exercício, a conversão de tipos deve ser resolvida usando um tipo específico de conversão.

Sua escolha será verificada durante a defesa.

# Capítulo IV IV
# Exercício 00: Conversão de escalar tipos

|  | Exercício<br>00 |
|---|---|
| Conversão de tipos<br>escalares | |
| Diretório de entrada : ez00/ | |
| Arquivos a serem entregues : Makef ile , * . cpp , * . (h , hpp} | |
| Funções permitidas: Qualquer função para converter de uma string para um int, um<br>float ou double. Isso ajudará, mas não fará todo o trabalho. | |

Escreva uma classe de cálculo ScalarConverter que conterá um método "convert" que recebe como parâmetro uma representação de string de um literal C++ em sua forma mais comum. Esse literal deve pertencer a um dos seguintes tipos de escalar:

- char
- int
- @Obt
- duplo

Exceto para parâmetros char, somente a notação decimal será usada.

Exemplos de char liberals: c , a' ,
Para simplificar as coisas, observe que os caracteres não exibíveis não devem ser usados como entradas. Se uma conversão para char não for exibível, será impressa uma mensagem informativa.

Exemplos de literais int: 0, -42, 42...

Exemplos de liberais de float: 0 . 0f, -4. 2f, 4 . 2f ...
Você precisa lidar com esses pseudo-liberais também (você sabe, para a ciência): -inf f, +inf I e nanf .

# Capítulo IV V

Exemplos de literais duplos: 0,0, -4,2, 4,2...
Você também precisa lidar com esses pseudo literais (você sabe, por diversão): -inf, +inf e nan.

Escreva um programa para testar se sua classe funciona conforme o esperado.

Primeiro, você precisa detectar o tipo do literal passado como parâmetro, convertê-lo de string para seu tipo real e, em seguida, convertê-lo explicitamente para os três outros tipos de dados. Por fim, exibir os resultados conforme mostrado abaixo.

Se uma conversão não fizer sentido ou transbordar, exiba uma mensagem para informar ao usuário que a conversão de tipos é impossível. Inclua qualquer cabeçalho necessário para lidar com limites numéricos e valores especiais.

# Capítulo VI V
# Exercício 01: Serialização

|  | Exercício: 01 |
|---|---|
| Serialização | |
| Diretório de entrada: ez01/ | |
| Arquivos a serem entregues : * . cpp , * . (h , hpp)<br>Nakef ile , | |
| Funções proibidas : Nenhuma | |

Implemente uma classe estática Serializer com os seguintes métodos:

`uintptr_t serialize(Data* ptr); Ele é um ponteiro de segundos e converte-o em um tipopeuintptr_t sem sinal`

Data* deserialize(uintptr t raw);
Ele pega um parâmetro não-significado de número inteiro e o converte em um ponteiro para Data.

Escreva um programa para testar se sua classe funciona conforme o esperado.

Você deve criar uma estrutura de dados não vazia (isso significa que ela tem membros de dados).

Use serial ize() no endereço do objeto Data e passe seu valor de retorno para deser i al ize () . Em seguida, verifique se o valor de retorno de deser i al ize () é igual ao ponteiro original.

Não se esqueça de entregar os arquivos de sua estrutura de dados.

# Capítulo VI VI

# Capítulo VI VII
# Exercício 02: Identificar o tipo real

| | |
|---|---|
|  | Exercício: 02 |
| Identificar o tipo real | |
| Diretório de entrada: ez02/ | |
| Arquivos para entregar: Nakef * . cpp , * . (h , hpp) i le , | |
| Funções proibidas : std: :t;ype inI o | |

Implemente uma classe Base que tenha apenas um destrutor virtual público. Crie três classes vazias A, B e C, que herdem publicamente da classe Base.

Essas quatro classes não precisam ser projetadas na Forma Canônica Ortodoxa.

Implementar as seguintes funções:

Base * generate (void) ;
Ele instancia aleatoriamente A, B ou C e retorna a instância como um ponteiro Base. Sinta-se à vontade para usar o que quiser para a implementação da escolha aleatória.

vo id i dent if y (Base* p) ;
Ele imprime o tipo real do objeto apontado por p: "A", "B" ou "C".

vo id i dent ify (Bas e& p) ;
Ela imprime o tipo real do objeto apontado por p: "A", "B" ou "C". É proibido usar um ponteiro dentro dessa função.

A inclusão do cabeçalho t ypeinf o é proibida.

Escreva um programa para testar se tudo funciona como esperado.