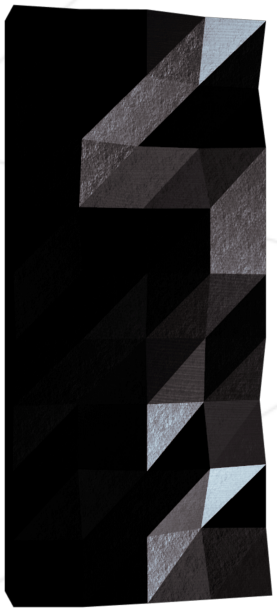

C++ - Módulo 06

Casts em C++

Preâmbulo: Este documento contém os exercícios para o Módulo 06 dos módulos de C++.

Versão: 8.0

Sumário

Capítulo I

Introdução

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

O objetivo destes módulos é apresentar a você a **Programação Orientada a Objetos**. Este será o ponto de partida de sua jornada em C++. Muitas linguagens são recomendadas para aprender OOP, mas escolhemos C++ pois ela é derivada de sua velha amiga, C. Como C++ é uma linguagem complexa, seu código aderirá ao padrão C++98 para manter as coisas simples.

Reconhecemos que o C++ moderno difere significativamente em muitos aspectos. Se você quer se tornar um desenvolvedor C++ proficiente, caberá a você explorar mais além do 42 Common Core!

Capítulo II

Regras gerais

Compilação

- Compile seu código com `c++` e as flags `-Wall -Wextra -Werror`
- Seu código ainda deve compilar se você adicionar a flag `-std=c++98`

Formatação e convenções de nomenclatura

- Os diretórios dos exercícios serão nomeados desta 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**. Arquivos contendo código de classe sempre serão nomeados de acordo com o nome da classe. Por exemplo:
  `NomeDaClasse.hpp/NomeDaClasse.h`, `NomeDaClasse.cpp`, ou `NomeDaClasse.tpp`.
  Então, se você tiver um arquivo de cabeçalho contendo a definição de uma classe "ParedeDeTijolo"representando uma parede de tijolos, seu nome será `ParedeDeTijolo.hpp`.
- A menos que especificado de outra forma, cada mensagem de saída deve terminar com um caractere de nova linha e ser exibida na saída padrão.
- *Adeus Norminette!* Nenhum estilo de codificação é imposto nos módulos C++. Você pode seguir o seu favorito. Mas lembre-se que o código que seus 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 programando em C. É hora de C++! Portanto:

- Você pode usar quase tudo da biblioteca padrão. Assim, em vez de se ater ao que você já conhece, seria inteligente usar as versões em C++ das funções C que você está acostumado o máximo possível.

- No entanto, você não pode usar nenhuma outra biblioteca externa. Isso significa que bibliotecas C++11 (e formas derivadas) e `Boost` são proibidas. As seguintes funções também são proibidas: `*printf()`, `*alloc()` e `free()`. Se você usá-las, sua nota será 0 e pronto.

- Observe que, a menos que explicitamente indicado de outra forma, as palavras-chave `using namespace <ns_name>` e `friend` são proibidas. Caso contrário, sua nota será -42.

- **Você só pode usar a STL nos Módulos 08 e 09.** Isso significa: nenhum **Contêiner** (vector/list/map, e assim por diante) e nenhum **Algoritmo** (qualquer coisa que exija a inclusão do cabeçalho `<algorithm>`) até lá. Caso contrário, sua nota será -42.

Alguns requisitos de design

- Vazamento de memória ocorre em C++ também. Quando você aloca memória (usando a palavra-chave `new`), você deve evitar **vazamentos de memória**.

- Do Módulo 02 ao Módulo 09, suas classes devem ser projetadas 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 de seus cabeçalhos independentemente de outros. Portanto, eles devem incluir todas as dependências de que precisam. No entanto, você deve evitar o problema de inclusão dupla adicionando **proteções 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, sinta-se à vontade para fazê-lo, desde que você 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! Sério, faça isso.

- Por Odin, por Thor! Use seu cérebro!!!

`Em relação ao Makefile para projetos C++, as mesmas regras do C se aplicam (consulte o capítulo Norma sobre o Makefile).`

Você terá que implementar muitas classes. Isso pode parecer tedioso, a menos que você seja capaz de criar scripts para seu editor de texto favorito.

Você tem uma certa liberdade para completar os exercícios. No entanto, siga as regras obrigatórias e não seja preguiçoso. Você perderia muitas informações úteis! Não hesite em ler sobre conceitos teóricos.

Capítulo III

Regra Adicional

A seguinte regra se aplica a todo o módulo e é obrigatória.

Para cada exercício, a conversão de tipo deve ser tratada usando um tipo específico de casting. Sua escolha será revisada durante a defesa.

Capítulo IV

Instruções de IA

Contexto

Este projeto foi desenvolvido para ajudá-lo a descobrir os blocos de construção fundamentais do seu treinamento em TIC.

Para ancorar adequadamente os conhecimentos e habilidades-chave, é essencial adotar uma abordagem criteriosa ao uso de ferramentas e suporte de IA.

A verdadeira aprendizagem fundamental exige um esforço intelectual genuíno — através de desafios, repetição e trocas de aprendizagem entre pares.

Para uma visão geral mais completa de nossa posição sobre a IA — como ferramenta de aprendizagem, como parte do currículo de TIC e como expectativa no mercado de trabalho — consulte as perguntas frequentes dedicadas na intranet.

Mensagem principal

- Construa bases sólidas sem atalhos.
- Desenvolva verdadeiramente habilidades técnicas e de poder.
- Experimente a verdadeira aprendizagem entre pares, comece a aprender como aprender e resolver novos problemas.
- A jornada de aprendizagem é mais importante que o resultado.
- Aprenda sobre os riscos associados à IA e desenvolva práticas eficazes de controle e contramedidas para evitar armadilhas comuns.

Regras para o aluno:

- Você deve aplicar o raciocínio às suas tarefas atribuídas, especialmente antes de recorrer à IA.
- Você não deve pedir respostas diretas à IA.
- Você deve aprender sobre a abordagem global da 42 em relação à IA.

Resultados da fase:

Nesta fase fundamental, você obterá os seguintes resultados:

- Obter bases sólidas em tecnologia e codificação.
- Saber por que e como a IA pode ser perigosa durante esta fase.

Comentários e exemplo:

- Sim, sabemos que a IA existe — e sim, ela pode resolver seus projetos. Mas você está aqui para aprender, não para provar que a IA aprendeu. Não perca seu tempo (nem o nosso) apenas para demonstrar que a IA pode resolver o problema dado.
- Aprender na 42 não é sobre saber a resposta — é sobre desenvolver a capacidade de encontrar uma. A IA lhe dá a resposta diretamente, mas isso o impede de construir seu próprio raciocínio. E o raciocínio leva tempo, esforço e envolve falhas. O caminho para o sucesso não deve ser fácil.
- Lembre-se de que durante os exames, a IA não estará disponível — sem internet, sem smartphones, etc. Você perceberá rapidamente se confiou demais na IA em seu processo de aprendizagem.
- A aprendizagem entre pares o expõe a diferentes ideias e abordagens, melhorando suas habilidades interpessoais e sua capacidade de pensar de forma divergente. Isso é muito mais valioso do que apenas conversar com um bot. Então não seja tímido — converse, faça perguntas e aprenda juntos!
- Sim, a IA fará parte do currículo — tanto como ferramenta de aprendizagem quanto como um tópico em si. Você até terá a chance de construir seu próprio software de IA. Para saber mais sobre nossa abordagem crescente, consulte a documentação disponível na intranet.

✓ Boa prática:

> Estou travado em um novo conceito. Pergunto a alguém próximo como ele abordou isso. Conversamos por 10 minutos — e de repente, clica. Entendi.

✗ Má prática:

> Uso secretamente a IA, copio algum código que parece certo. Durante a avaliação entre pares, não consigo explicar nada. Eu falho. Durante o exame — sem IA — estou travado novamente. Eu falho.

Capítulo V

Exercício 00: Conversão de tipos escalares

	Exercício : 00
Conversão de tipos escalares	
Pasta de entrega : $ex00/$	
Arquivos para entregar : `Makefile, *.cpp, *.{h, hpp}`	
Funções ou bibliotecas autorizadas : `Qualquer função para converter de uma string para um int, um float, ou um double. Isso ajudará, mas não fará todo o trabalho.`	

Escreva uma classe ScalarConverter que conterá apenas um método `static` "convert"que receberá como parâmetro uma representação em string de um literal C++ em sua forma mais comum e exibirá seu valor na seguinte série de tipos escalares:

- char
- int
- float
- double

Como esta classe não precisa armazenar nada, ela não deve ser instanciável por usuários.

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

Exemplos de literais char: `'c'`, `'a'`, ...
Para simplificar, observe que caracteres não exibíveis não devem ser usados como entradas. Se uma conversão para char não for exibível, imprima uma mensagem informativa.

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

Exemplos de literais float: `0.0f`, `-4.2f`, `4.2f`...
Você tem que lidar com esses pseudo-literais também (você sabe, pela ciência): `-inff`,

`+inff`, e `nanf`.

Exemplos de literais double: `0.0`, `-4.2`, `4.2`...
Você tem que lidar com esses pseudo-literais também (você sabe, por diversão): `-inf`, `+inf`, e `nan`.

Escreva um programa para testar se sua classe funciona como esperado.

Você deve primeiro detectar o tipo do literal passado como parâmetro, convertê-lo da string para seu tipo real e, em seguida, convertê-lo **explicitamente** para os outros três tipos de dados. Por fim, exiba os resultados conforme mostrado abaixo.

Se uma conversão não fizer sentido ou causar overflow, exiba uma mensagem para informar ao usuário que a conversão de tipo é impossível. Inclua qualquer cabeçalho que você precise para lidar com limites numéricos e valores especiais.

```
./convert 0
char: Não exibível
int: 0
float: 0.0f
double: 0.0
./convert nan
char: impossível
int: impossível
float: nanf
double: nan
./convert 42.0f
char: '*'
int: 42
float: 42.0f
double: 42.0
```

Capítulo VI

Exercício 01: Serialização

	Exercice : 01
Serialização	
Pasta de entrega : $ex01/$	
Arquivos para entregar : `Makefile, *.cpp, *.{h, hpp}`	
Funções não permitidas : `Nenhuma`	

Implemente uma classe Serializer, que não será inicializável pelo usuário de forma alguma, com os seguintes métodos `static`:

`uintptr_t serialize(Data* ptr);`
Recebe um ponteiro e o converte para o tipo inteiro não assinado `uintptr_t`.

`Data* deserialize(uintptr_t raw);`
Recebe um parâmetro inteiro não assinado e o converte em um ponteiro para `Data`.

Escreva um programa para testar se sua classe funciona como esperado.

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

Use `serialize()` no endereço do objeto Data e passe seu valor de retorno para `deserialize()`. Em seguida, garanta que o valor de retorno de `deserialize()` seja igual ao ponteiro original.

Não se esqueça de entregar os arquivos de sua estrutura `Data`.

Capítulo VII

Exercício 02: Identificar tipo real

	Exercice : 02
Identificar tipo real	
Pasta de entrega : $ex02/$	
Arquivos para entregar : `Makefile, *.cpp, *.{h, hpp}`	
Funções não permitidas : `std::typeinfo`	

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

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

Implemente as seguintes funções:

```
Base * generate(void);
```

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 de escolha aleatória.

```
void identify(Base* p);
```

Imprime o tipo real do objeto apontado por p: "A", "B"ou "C".

```
void identify(Base& p);
```

Imprime o tipo real do objeto referenciado por p: "A", "B"ou "C". Usar um ponteiro dentro desta função é proibido.

Incluir o cabeçalho `typeinfo` é proibido.

Escreva um programa para testar se tudo funciona como esperado.

Capítulo VIII

Entrega e Avaliação por Pares

Entregue seu projeto no seu repositório `Git` como de costume. Apenas o trabalho dentro de seu repositório será avaliado durante a defesa. Não hesite em verificar novamente os nomes de suas pastas e arquivos para garantir que eles estão corretos.

Durante a avaliação, uma breve **modificação do projeto** pode ser ocasionalmente solicitada. Isso pode envolver uma pequena mudança de comportamento, algumas linhas de código para escrever ou reescrever, ou um recurso fácil de adicionar.

Embora esta etapa possa **não ser aplicável a todos os projetos**, você deve estar preparado para ela se for mencionada nas diretrizes de avaliação.

Esta etapa visa verificar sua compreensão real de uma parte específica do projeto. A modificação pode ser realizada em qualquer ambiente de desenvolvimento que você escolher (por exemplo, sua configuração usual), e deve ser viável em poucos minutos — a menos que um prazo específico seja definido como parte da avaliação.
Você pode, por exemplo, ser solicitado a fazer uma pequena atualização em uma função ou script, modificar uma exibição ou ajustar uma estrutura de dados para armazenar novas informações, etc.

Os detalhes (escopo, alvo, etc.) serão especificados nas **diretrizes de avaliação** e podem variar de uma avaliação para outra para o mesmo projeto.

16D85ACC441674FBA2DF65190663E136253996A5020347143B460E2CF3A3784D794B
104265933C3BE5B62C4E062601EC8DD1F82FEB73CB17AC57D49054A7C29B5A5C1D8
2027A997A3E24E387