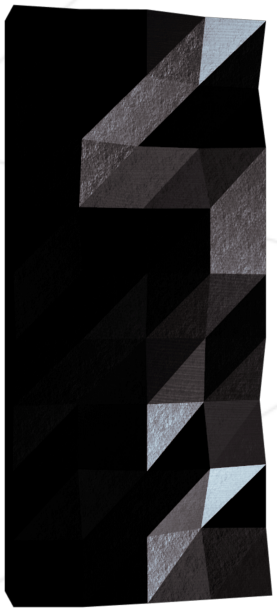

C++ - Módulo 03

Herança

Preâmbulo:
Este documento contém os exercícios do Módulo 03 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, ou "C com Classes"(fonte: Wikipedia).

O objetivo destes módulos é introduzi-lo à **Programação Orientada a Objetos**. Este será o ponto de partida da sua jornada em C++. Muitas linguagens são recomendadas para aprender OOP, mas decidimos escolher C++ porque é derivada da sua velha amiga C. Como esta é uma linguagem complexa, e para manter as coisas simples, o seu código estará em conformidade com o padrão C++98.

Estamos cientes de que o C++ moderno é muito diferente em muitos aspetos. Então, se você quer se tornar um desenvolvedor C++ proficiente, cabe a você ir mais longe após o 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

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 IV

Exercício 00: Eeeee... ABRIU!

	Exercice : 00
Eeeee... ABRIU!	
Pasta de entrega : $ex00/$	
Arquivos para entregar : `Makefile, main.cpp, ClapTrap.{h, hpp}, ClapTrap.cpp`	
Funções não permitidas : `Nenhuma`	

Primeiro, você tem que implementar uma classe! Que original!

Ela será chamada **ClapTrap** e terá os seguintes atributos privados inicializados com os valores especificados entre parênteses:

- Name, que é passado como parâmetro para o construtor
- Hit points (10), representando a saúde do ClapTrap
- Energy points (10)
- Attack damage (0)

Adicione as seguintes funções membro públicas para que o ClapTrap se comporte de forma mais realista:

- `void attack(const std::string& target);`
- `void takeDamage(unsigned int amount);`
- `void beRepaired(unsigned int amount);`

Quando ClapTrap ataca, ele faz com que seu alvo perca `<attack damage>` pontos de vida. Quando ClapTrap se repara, ele recupera `<amount>` pontos de vida. Atacar e reparar custam 1 ponto de energia cada. Claro, ClapTrap não pode fazer nada se não tiver pontos de vida ou energia sobrando. No entanto, como estes exercícios servem como uma introdução, as instâncias de ClapTrap não devem interagir diretamente umas com as outras, e os parâmetros não se referirão a outra instância de ClapTrap.

Em todas estas funções membro, você precisa imprimir uma mensagem para descrever o que acontece. Por exemplo, a função `attack()` pode exibir algo como (claro, sem os colchetes angulares):

```
ClapTrap <name> attacks <target>, causing <damage> points of damage!
```

Os construtores e o destrutor também devem exibir uma mensagem, para que seus avaliadores possam ver facilmente que eles foram chamados.

Implemente e entregue seus próprios testes para garantir que seu código funcione como esperado.

Capítulo V

Exercício 01: Serena, my love!

	Exercice : 01
Serena, my love!	
Pasta de entrega : *ex*01/	
Arquivos para entregar : `Arquivos do exercício anterior + ScavTrap.{h, hpp}, ScavTrap.cpp`	
Funções não permitidas : `Nenhuma`	

Como você nunca pode ter ClapTraps suficientes, você agora criará um robô derivado. Ele será chamado **ScavTrap** e herdará os construtores e destrutor de ClapTrap. No entanto, seus construtores, destrutor e `attack()` imprimirão mensagens diferentes. Afinal, os ClapTraps estão cientes de sua individualidade.

Note que o encadeamento adequado de construção/destruição deve ser mostrado em seus testes. Quando um ScavTrap é criado, o programa começa construindo um ClapTrap. A destruição ocorre na ordem inversa. Por quê?

ScavTrap usará os atributos de ClapTrap (atualize ClapTrap de acordo) e deve inicializá-los para:

- Name, que é passado como parâmetro para o construtor
- Hit points (100), representando a saúde do ClapTrap
- Energy points (50)
- Attack damage (20)

ScavTrap também terá sua própria habilidade especial:

```
void guardGate();
```

Esta função membro exibirá uma mensagem indicando que ScavTrap está agora no modo de porteiro.

Não se esqueça de adicionar mais testes ao seu programa.

Capítulo VI

Exercício 02: Trabalho repetitivo

	Exercice : 02
Trabalho repetitivo	
Pasta de entrega : $ex02/$	
Arquivos para entregar : `Arquivos dos exercícios anteriores + FragTrap.{h, hpp}, FragTrap.cpp`	
Funções não permitidas : `Nenhuma`	

Fazer ClapTraps provavelmente está começando a te irritar.

Agora, implemente uma classe **FragTrap** que herda de ClapTrap. É muito semelhante ao ScavTrap. No entanto, suas mensagens de construção e destruição devem ser diferentes. O encadeamento adequado de construção/destruição deve ser mostrado em seus testes. Quando um FragTrap é criado, o programa começa construindo um ClapTrap. A destruição ocorre na ordem inversa.

O mesmo vale para os atributos, mas com valores diferentes desta vez:

- Name, que é passado como parâmetro para o construtor
- Hit points (100), representando a saúde do ClapTrap
- Energy points (100)
- Attack damage (30)

FragTrap também tem uma habilidade especial:

```
void highFivesGuys(void);
```

Esta função membro exibe um pedido de high-five positivo na saída padrão.

Novamente, adicione mais testes ao seu programa.

Capítulo VII

Exercício 03: Agora está estranho!

	Exercice : 03
Agora está estranho!	
Pasta de entrega : $ex03/$	
Arquivos para entregar : `Arquivos dos exercícios anteriores + DiamondTrap.{h, hpp}, DiamondTrap.cpp`	
Funções não permitidas : `Nenhuma`	

Neste exercício, você criará um monstro: um ClapTrap que é metade FragTrap, metade ScavTrap. Ele será chamado **DiamondTrap**, e herdará de ambos FragTrap E ScavTrap. Isso é tão arriscado!

A classe DiamondTrap terá um atributo privado chamado `name`. Este atributo deve ter exatamente o mesmo nome de variável que na classe base ClapTrap (sem se referir ao nome do robô).

Para ser mais claro, aqui estão dois exemplos:
Se a variável de ClapTrap for `name`, dê à variável de DiamondTrap o nome `name`.
Se a variável de ClapTrap for `_name`, dê à variável de DiamondTrap o nome `_name`.

Seus atributos e funções membro serão herdados de suas classes pai:

- Name, que é passado como um parâmetro para um construtor
- `ClapTrap::name` (parâmetro do construtor + sufixo `"_clap_name"`)
- Hit points (FragTrap)
- Energy points (ScavTrap)
- Attack damage (FragTrap)
- `attack()` (ScavTrap)

Além das funções especiais de ambas as classes pai, DiamondTrap terá sua própria habilidade especial:

```
void whoAmI();
```

Esta função membro exibirá tanto seu nome quanto seu nome ClapTrap.

Claro, a instância ClapTrap de DiamondTrap será criada uma vez, e apenas uma vez. Sim, há um truque.

Novamente, adicione mais testes ao seu programa.

Você conhece os flags de compilador -Wshadow e -Wno-shadow?

Você pode passar neste módulo sem completar o exercício 03.

Capítulo VIII

Entrega e Avaliação por Pares

Envie sua tarefa em 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 estejam 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.

```
???????????  XXXXXXXXXX = $3$$cf36316f07f871b4f14926007c37d388
```