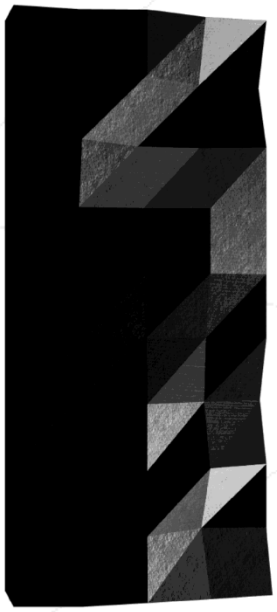

C++ - Módulo 05

Repetição e exceções

Summary:

Este documento contém exercícios clínicos do Módulo 05 dos módulos Ce módulos.

Conteúdo

Capítulo I 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 III

Exercício 00: Mamãe, quando eu crescer Quero ser um burocrata!

	Exercício : 00
	Mamãe, quando eu crescer, quero ser um burocrata!
	Turnindirectory: ei00/
	Filestoturnin : Makefile, main.cpp, Bureaucrat.{h, hpp}, Bureaucrat.cpp
	Funções proibidas: Nenhuma

Observe que as aulas de exceção não precisam ser elaboradas na Forma Canônica Ortodoxa. Mas todas as outras classes precisam.

Vamos projetar um pesadelo artificial de escritórios, corredores, formulários e filas de espera. Parece divertido? Não? Que pena.

Primeiro, comece pela menor engrenagem dessa vasta máquina burocrática: o

burocrata. Um burocrata deve ter:

- Um nome constante.
- E uma nota que varia de 1 (nota mais alta possível) a 150 (nota mais baixa possível).

Qualquer tentativa de instanciar um burocrata usando uma nota inválida deve gerar uma exceção:
ou um Burocrata : : GradeTooHighExcep9ion ou um Bureaucrat : : GradeTooLovExcep9ion.

Você fornecerá get ters para esses dois atributos: get Name () e get Grade (). Implemente também duas funções de membro para incrementar ou decrementar a nota do burocrata. Se a nota estiver fora do intervalo, ambas lançarão as mesmas exceções que o construtor.

Lembre-se. Como a nota 1 é a mais alta e a 150 a mais baixa, o incremento de uma nota 3 deve dar uma nota 2 ao burocrata.

As exceções lançadas devem ser capturáveis usando blocos try e catch:

Você implementará uma sobrecarga do operador insertion (") para imprimir algo como (sem os colchetes angulares):

4nameâ , burocrata grau 4grade .

Como de costume, faça alguns testes para comprovar que tudo funciona conforme o esperado.

Capítulo IV

Exercício 01: Formem-se, larvas!

	Exercício: 01
Formem-se, vermes!	
Diretório de entrada: ez01/	
Arquivos para entrega: Arquivos do exercício anterior + Form . (h , hpp) , Form . cpp	
Funções proibidas : Nenhuma	

Agora que você tem burocratas, vamos dar a eles algo para fazer. Que atividade poderia ser melhor do que preencher uma pilha de formulários?

Em seguida, vamos criar uma classe Form. Ela tem:

- Um nome constante.
- Um booleano que indica se ele é assinado (na construção, não é).
- É necessário um grau constante para assiná-lo.
- E uma nota constante necessária para executá-lo.

Todos esses atributos são privados, não protegidos.

As notas do Formulário seguem as mesmas regras que se aplicam ao Burocrata. Portanto, as seguintes exceções serão lançadas se uma nota de formulário estiver fora dos limites: Formulário : : GradeTooH i ghExcept ion e Forn : : GradeTooLovExcept ion.

Da mesma forma que antes, escreva getters para todos os atributos e uma sobrecarga do operador de inserção (") que imprime todas as informações do formulário.

Adicione também uma função membro bes igned () ao Formulário que recebe um Burocrata como parâmetro. Ela altera o status do formulário para assinado se a nota do burocrata for alta o suficiente (maior ou igual à exigida). Lembre-se de que o grau 1 é maior que o grau 2.
Se a nota for muito baixa, lance um Form : : GradeTooLovExcept ion.

Por fim, adicione uma função membro s i gnForn() ao Bureaucrat. Se o formulário for assinado, ele imprimirá algo como:

```
<bureaucrat> assinou <form>
```

Caso contrário, ele imprimirá algo como:

```
<burocrata> não pôde assinar o 'formulário> porque <razão>
```

Implemente e entregue alguns testes para garantir que tudo funcione conforme o esperado.

Capítulo V

Exercício 02: Não, você precisa do formulário 28B, não do 28C...

	Exercício: 02
Não, você precisa do formulário 28B, não do 28C...	
Diretório de entrada: ez02/	
Arquivos a serem entregues : Makef ile , main . cpp , B u r e a u c r a t . f(h , hpp}, cppd , Bureaucrat . cpp + AForm . f(h , hpp}, cpp? , ShrubberyCreat i onForm . f(h , hpp) , cpp) , + Robot;omyRequestForn . I(h , hpp) , cppd , Pres ident i alPardonForm . [(h , hpp) , cpp)	
Funções proibidas : Nenhuma	

Como agora você tem formulários básicos, é hora de criar mais alguns que realmente façam alguma coisa.

Em todos os casos, a classe base Form deve ser uma classe abstrata e, portanto, deve ser renomeada para AForm. Lembre-se de que os atributos do formulário precisam permanecer privados e que eles estão na classe base.

Adicione as seguintes classes concretas:

- ShrubberyCreationForm: Graus necessários: sinal 145, exec 137
  Cria um arquivo tt arget b_shrubbery no diretório de trabalho e grava árvores ASCII dentro dele.
- RobotomyRequestForm: Notas necessárias: sign 72, exec 45
  Faz alguns ruídos de perfuração. Em seguida, informa que o alvo foi robotizada com sucesso em 50% das vezes. Caso contrário, informa que a robotização falhou.
- PresidentialPardonForm: Notas necessárias: sign 25, exec 5
  Informa que 4target foi perdoado por Zaphod Beeblebrox.

Todos eles recebem apenas um parâmetro em seu construtor: o destino do formulário. Por exemplo, "home" se você quiser plantar arbustos em casa.

Agora, adicione a função membro execute (Bureaucrat const & executor) const ao formulário base e implemente uma função para executar a ação do formulário das classes concretas. Você precisa verificar se o formulário está assinado e se o grau do burocrata que está tentando executar o formulário é alto o suficiente. Caso contrário, lance uma exceção apropriada.

Você decide se deseja verificar os requisitos em cada classe concreta ou na classe base (e depois chamar outra função para executar o formulário). Entretanto, uma maneira é mais bonita do que a outra.

Por fim, adicione a função membro executeForn(Form const & I ore) ao Bureau-cat. Ela deve tentar executar o formulário. Se for bem-sucedida, imprima algo como:

```
<bureaucrat> executado <form>
```

Caso contrário, imprima uma mensagem de erro explícita.

Implemente e entregue alguns testes para garantir que tudo funcione conforme o esperado.

Capítulo VI

Exercício 03: Pelo menos isso bate preparo de café

	Exercício : 03
Atestthsbeasco|Re-making	
Turnindiecory: ei03/	
Arquivos de exercícios anteriores Arquivos de exercícios anteriores + Intern.{h, hpp}, Intern.cpp	
Funções proibidas: Nenhuma	

Como o preenchimento de formulários já é irritante o suficiente, seria cruel pedir aos nossos burocratas que fizessem isso o dia todo. Felizmente, existem estagiários. Neste exercício, você deve implementar a classe Intern. O estagiário não tem nome, nem nota, nem características exclusivas. A única coisa com que os burocratas se importam é que eles façam seu trabalho.

No entanto, o intern tem uma capacidade importante: a função nakeForn( ). Ela recebe duas cadeias de caracteres. A primeira é o nome de um formulário e a segunda é o destino do formulário. Ela retorna um ponteiro para um objeto Form (cujo nome é o que foi passado como parâmetro) cujo destino será inicializado com o segundo parâmetro.

Ele imprimirá algo como:

```
O estagiário cria <form'
```

Se o nome do formulário passado como parâmetro não existir, imprima uma mensagem de erro explícita.

Você deve evitar soluções ilegíveis e feias, como o uso de uma floresta if/elseif/else. Esse tipo de coisa não será aceito durante o processo de avaliação. Você não está mais na Piscine (piscina). Como de costume, você precisa testar se tudo funciona conforme o esperado.

Por exemplo, o código abaixo cria um RobotomyRequestForm direcionado a "Bender":