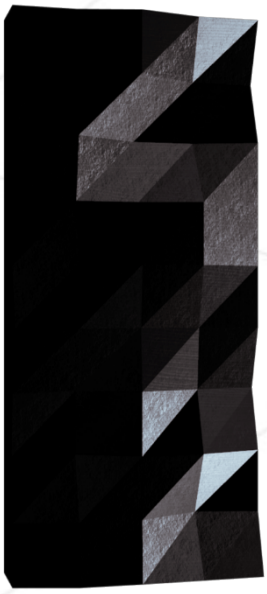
DeepL
Assine o DeepL Pro para traduzir arquivos maiores.
Mais informações em www.DeepL.com/pro.



C++ - Módulo 00

Namespaces, classes, funções dos membros, stdio streams, listas de inicialização, estática, const, e algumas outras coisas básicas

Resumo:
Este documento contém os exercícios do Módulo 00 dos módulos C++.

Versão: 7

Conteúdo

Capítulo I

Introdução

C++ é uma linguagem de programação de propósito geral criada por Bjarne Stroustrup como uma ex- tensão da linguagem de programação C, ou "C com Classes" (fonte: Wikipedia).

O objetivo destes módulos é apresentar-lhe a **Programação Orientada a Objetos**. Este será o ponto de partida de sua jornada C++. Muitas linguagens são recomendadas para aprender o OOP. Decidimos escolher C++, pois é derivado de seu velho amigo C. Como esta é uma linguagem complexa, e para manter as coisas simples, seu código estará de acordo com o padrão C++98.

Estamos conscientes de que o C++ moderno é muito diferente em muitos aspectos. Portanto, se você quer se tornar um desenvolvedor C++ proficiente, cabe a você ir mais além após o 42 Common Core!

Você descobrirá novos conceitos passo a passo. Os exercícios aumentarão progressivamente em complexidade.

Capítulo II Regras gerais

Compilação

- Compile seu código com c++ e as bandeiras -Wall -Wextra -Werror
- Seu código ainda deve ser compilado se você adicionar a bandeira -std=c+++98

Formatação e convenções de nomenclatura

- Os diretórios de exercícios serão nomeados desta forma: ex00, ex01, ... , exn
- Nomeie seus arquivos, classes, funções, funções dos membros e atributos, conforme exigido nas diretrizes.
- Escreva os nomes das classes no formato **UpperCamelCase.** Arquivos contendo código de classe serão sempre nomeados de acordo com o nome da classe. Por exemplo, os arquivos com o código de classe serão sempre nomeados de acordo com o nome da classe: ClassName.hpp/ClassName.h, ClassName.cpp, ou ClassName.tpp. Então, se você tiver um arquivo de cabeçalho contendo a definição de uma classe "BrickWall" representando uma parede de tijolo, seu nome será BrickWall.hpp.
- A menos que especificado de outra forma, todas as mensagens de saída devem ser terminadas por um novo caractere de linha e exibidas na saída padrão.
- *Adeus Norminette!* Nenhum estilo de codificação é aplicado nos módulos C++. Você pode seguir seu favorito. Mas tenha em mente que um código que seus colegas avaliadores não conseguem entender é um código que eles não conseguem avaliar. Faça seu melhor para escrever um código limpo e legível.

Permitido/Proibido

Você não está mais codificando em C. Tempo para C++! Portanto:

- Você tem permissão para usar quase tudo da biblioteca padrão. Assim, ao invés de se ater ao que você já sabe, seria inteligente usar o máximo possível as versões C++-ish das funções C às quais você está acostumado.
- Entretanto, não é possível utilizar nenhuma outra biblioteca externa. Isso significa que as 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.

- Note que, a menos que explicitamente declarado de outra forma, o uso do namespace <ns_name> e
  são proibidas as palavras-chave de amigos. Caso contrário, sua nota será -42.
- **Você está autorizado a usar o STL somente no Módulo 08.** Isso significa: nenhum **Container** (vetor/lista/mapa/e assim por diante) e nenhum **Algoritmo** (qualquer coisa que requeira incluir o <algoritmo> cabeçalho) até lá. Caso contrário, sua nota será de -42.

Alguns requisitos de projeto

- 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 08, suas aulas devem ser projetadas no **Formulário Canônico Ortodoxo, exceto quando explícitamente declarado em contrário.**
- Qualquer implementação de função colocada em um arquivo de cabeçalho (exceto para os modelos de função) significa 0 para o exercício.
- Você deve ser capaz de usar cada um de seus cabeçalhos independentemente dos outros. Assim, eles devem incluir todas as dependências de que precisam. Entretanto, você deve evitar o problema da dupla inclusão, acrescentando **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 estas atribuições não são verificadas por um programa, sinta-se à vontade para fazê-lo desde que você entregue os arquivos obrigatórios.
- s vezes, as diretrizes de um exercício parecem curtas, mas os exemplos podem mostrar exigências que não estão explicitamente escritas nas instruções.
- Leia cada módulo completamente antes de começar! Realmente, faça-o.
- Por Odin, por Thor! Use seu cérebro!!!

Você terá que implementar muitas aulas. Isto pode parecer entediante, a menos que você seja capaz de escrever seu editor de texto favorito.

É-lhe dada uma certa liberdade para completar 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: Megafone

Exercício: 00 Megafone

Diretório de entrada : *ex00/*

Arquivos para entregar : `Makefile, megafone.cpp`

Funções proibidas : Nenhuma

Só para garantir que todos estejam acordados, escreva um programa que se comporte da seguinte forma:

```
$>./megafone "shhhhh... Acho que os alunos estão dormindo..."
SHHHHH... ACHO QUE OS ALUNOS ESTÃO DORMINDO...
$>./megafone Damnit " ! ""Desculpem estudantes, pensei que esta coisa estava
desligada". DAMNIT ! DESCULPEM ESTUDANTES, EU PENSEI QUE ISTO ESTAVA DESLIGADO.
$>./megafone
* RUÍDO DE FEEDBACK ALTO E INSUPORTÁVEL *
$>
```

Solucionar os exercícios de maneira C++.

Capítulo IV

Exercício 01: Minha lista telefônica fantástica

	Exercício : 01
Minha lista telefônica incrível	
Diretório de entrada : *ex01/*	
Arquivos para entregar : `Makefile, *.cpp, *.{h, hpp}`	
Funções proibidas : `Nenhuma`	

Bem-vindo aos anos 80 e sua tecnologia inacreditável! Escreva um programa que se comporta como um software de lista telefônica incrível.

Você tem que implementar duas classes:

- **PhoneBook**
  - Ela tem uma série de contatos.
  - Ele pode armazenar um máximo de **8 contatos**. Se o usuário tentar adicionar um 9º contato, substitua o mais antigo pelo novo.
  - Por favor, note que a alocação dinâmica é proibida.
- **Contato**
  - Representa um contato telefônico.

Em seu código, a lista telefônica deve ser instanciada como uma instância da classe **PhoneBook. A** mesma coisa para os contatos. Cada um deles deve ser instanciado como uma instância da classe **Contato**. Você é livre para desenhar as classes como quiser, mas tenha em mente que tudo que será sempre usado dentro de uma classe é privado, e que tudo que pode ser usado fora de uma classe é público.

`Não se esqueça de assistir aos vídeos da intranet.`

Ao iniciar o programa, a lista telefônica está vazia e o usuário é solicitado a digitar um dos três comandos. O programa aceita apenas ADD, SEARCH e EXIT.

- **ADD**: salvar um novo contato
  - Se o usuário entra neste comando, ele é solicitado a inserir as informações do novo campo de contato, um de cada vez. Uma vez que todos os campos tenham sido preenchidos, acrescente o contato à lista telefônica.
  - Os campos de contato são: primeiro nome, sobrenome, apelido, número de telefone e segredo mais obscuro. Um contato salvo não pode ter campos vazios.
- **PESQUISA**: exibir um contato específico
  - Exibir os contatos salvos como uma lista de **4 colunas**: índice, primeiro nome, sobrenome e apelido.
  - Cada coluna deve ter **10 caracteres** de largura. Um caractere de tubo ('|') os separa. O texto deve estar alinhado à direita. Se o texto for mais longo que a coluna, deve ser truncado e o último caractere exibível deve ser substituído por um ponto ('.').
  - Em seguida, solicite novamente ao usuário o índice da entrada a ser exibida. Se o índice estiver fora do alcance ou errado, defina um comportamento relevante. Caso contrário, exibir as informações de contato, um campo por linha.
- **SAÍDA**
  - O programa desiste e os contatos são perdidos para sempre!
- **Qualquer outra entrada é descartada.**

Uma vez que um comando tenha sido executado corretamente, o programa espera por outro. Ele pára quando o usuário entra EXIT.

Dê um nome relevante ao seu executável.

http://www.cplusplus.com/reference/string/string/ e, é claro, http://www.cplusplus.com/reference/iomanip/

Capítulo V

Exercício 02: O trabalho de seus sonhos

	Exercício : 02
O trabalho de seus sonhos	
Diretório de entrada : *ex02/*	
Arquivos para entregar : `Makefile, Account.cpp, Account.hpp, testes.cpp`	
Funções proibidas : `Nenhuma`	

`Account.hpp, tests.cpp e o arquivo de log estão disponíveis para download na página da intranet do módulo.`

Hoje é o seu primeiro dia na *GlobalBanksters United*. Depois de passar com sucesso nos testes de re-cruzamento (graças a alguns truques do *Microsoft Office que* um amigo lhe mostrou), você se juntou à equipe de desenvolvimento. Você também sabe que o recrutador ficou surpreso com a rapidez com que você instalou o *Adobe Reader*. Esse pequeno extra fez toda a diferença e o ajudou a derrotar todos os seus adversários (também conhecidos como os outros candidatos): você conseguiu!

De qualquer forma, seu gerente acabou de lhe dar algum trabalho a fazer. Sua primeira tarefa é recriar um arquivo perdido. Algo deu errado e um arquivo fonte foi apagado por engano. Infelizmente, seus colegas não sabem o que é Git e usam chaves USB para compartilhar código. Neste ponto, faria sentido sair deste lugar agora mesmo. No entanto, você decide ficar. Desafio aceito!

Seus colegas desenvolvedores lhe dão um monte de arquivos. A compilação de testes.cpp revela que o arquivo que falta é Account.cpp. Por sorte, o arquivo de cabeçalho Account.hpp foi salvo. Há também um arquivo de log. Talvez você possa usá-lo para entender como a classe **Account** foi implementada.

Você começa a recriar o arquivo Account.cpp. Em apenas alguns minutos, você codifica algumas linhas de C++ puro e impressionante. Após algumas compilações fracassadas, seu programa passa nos testes. Sua saída corresponde perfeitamente àquela salva no arquivo de log (**exceto pelos timestamps** que obviamente serão diferentes já que os testes salvos no arquivo de log foram executados antes de você ser contratado).

Maldição, você é impressionante!

Você pode passar este módulo sem fazer o exercício 02.