



C++ - Módulo 00

Espaços para nomes, classes, funções de membro, fluxos stdio, listas de inicialização, static, const e algumas outras coisas básicas

Resumo:

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

Versão: 9.1

Conteúdo

I	Introdução	2
II	Regras gerais	3
III	Exercício 00: Megafone	6
IV	Exercício 01: Minha Incrível Agenda Telefônica	7
	Exercício 02: O emprego dos seus sonhos	9
VI	Submissão e avaliação por pares	11

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 apresentar a **Programação Orientada a Objetos**.

Este será o ponto de partida da sua jornada em C++. Muitas linguagens são recomendadas para aprender POO. Decidimos escolher C++ por ser derivada da sua velha amiga C.

Como C++ é uma linguagem complexa, e para manter as coisas simples, seu código seguirá o padrão C++98.

Sabemos que o C++ moderno é muito diferente em muitos aspectos. Então, se você quer se tornar um desenvolvedor C++ proficiente, cabe a você ir além, além dos 42 Common Core!

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

Capítulo II

Regras gerais

Compilando

- Compile seu código com c++ e os sinalizadores -Wall -Wextra -Werror
- Seu código ainda deve compilar se você adicionar o sinalizador -std=c++98

Convenções de formatação e nomenclatura

- Os diretórios dos exercícios serão nomeados desta forma: ex00, ex01, ... , exn
- Nomeie seus arquivos, classes, funções, funções de membro e atributos conforme necessário em as 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: ClassName.hpp/ClassName.h, ClassName.cpp ou ClassName.hpp. Então, se você tiver um arquivo de cabeçalho contendo a definição da classe "BrickWall", que representa uma parede de tijolos, seu nome será BrickWall.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 de que código que seus avaliadores não conseguem entender é código que eles não podem avaliar. Faça o possível para escrever um código limpo e legível.

Permitido/Proibido

Você não está mais programando em C. Hora de C++! Portanto:

- Você pode usar quase tudo da biblioteca padrão. Portanto, em vez de se ater ao que você já conhece, seria inteligente usar as versões em C++ das funções C com as quais você está acostumado, sempre que possível.
- No entanto, você não pode usar nenhuma outra biblioteca externa. Isso significa que as bibliotecas C++11 (e 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 o contrário, o uso do namespace <ns_name> e Palavras-chave de amigos são proibidas. Caso contrário, sua nota será -42.
- **Você só pode usar o STL nos Módulos 08 e 09.** Isso significa que não há **Contêineres** (vetor/lista/mapa e assim por diante) nem **Algoritmos** (qualquer coisa que exija a inclusão do cabeçalho <algoritmo>) até lá. Caso contrário, sua nota será -42.

Alguns requisitos de design

- 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 09, suas aulas devem ser elaboradas na Igreja **Ortodoxa** **Forma canônica, exceto quando explicitamente declarado 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 conseguir usar cada um dos seus cabeçalhos independentemente dos outros. Portanto, eles devem incluir todas as dependências necessárias. 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 necessário (por exemplo, para dividir seu código). Como essas atribuições não são verificadas por um programa, sintá-se à vontade para fazê-lo, 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! Sério, faça isso.
- Por Odin, por Thor! Use seu cérebro!!!



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



Você terá que implementar muitas classes. Isso pode parecer tedioso, a menos que você saiba programar 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 iria
perca muitas informações úteis! Não hesite em ler sobre
conceitos teóricos.

Capítulo III

Exercício 00: Megafone

	Exercício: 00
Megafone	
Diretório de entrega: ex00/	
Arquivos para entrega: Makefile, megaphone.cpp Funções	
proibidas: Nenhuma	

Só para ter certeza de que todos estão acordados, escreva um programa que produza a seguinte saída:


```
$>./megafone "shhhhh... Acho que os alunos estão dormindo..."
SHHHHH... ACHO QUE OS ALUNOS ESTÃO DORMINDO... $>./megafone Droga, DROGA!
DESCULPE, ALUNOS, PENSEI QUE " " ! "Desculpe, alunos, pensei que isso fosse estranho."
ESTA COISA ESTIVESSE DESLIGADA. $>./megafone " RÚIDO DE FEEDBACK ALTO E INSUPORTÁVEL " $>
```



Resolva os exercícios no estilo C++.

Capítulo IV

Exercício 01: Minha Incrível Lista telefônica

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

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

Você precisa implementar duas classes:

- **Lista telefônica**

- Possui uma variedade de contatos.
- É possível armazenar no máximo **8 contatos**. Caso o usuário tente adicionar um 9º contato, o mais antigo será substituído pelo novo.
- Observe que a alocação dinâmica é proibida.

- **Contato**

- Representa um contato da lista telefônica.

No seu código, a lista telefônica deve ser instanciada como uma instância da classe **PhoneBook**. O mesmo vale para os contatos. Cada um deles deve ser instanciado como uma instância da classe **Contact**. Você é livre para criar as classes como quiser, mas lembre-se de que tudo o que sempre será usado dentro de uma classe é privado e tudo o que pode ser usado fora de uma classe é público.



Não se esqueça de assistir aos vídeos da intranet.

Na inicialização do programa, a lista telefônica está vazia e o usuário é solicitado a inserir um de três comandos. O programa aceita apenas ADD, SEARCH e EXIT.

- **ADICIONAR:** salvar um novo contato

• Se o usuário digitar este comando, será solicitado a inserir as informações do novo contato, um campo de cada vez. Após o preenchimento de todos os campos, adicione o contato à agenda.

• Os campos de contato são: 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, nome, sobrenome nome e apelido.

• Cada coluna deve ter **10 caracteres** de largura. Uma barra vertical ('|') as separa. O texto deve ser alinhado à direita. Se o texto for maior que a coluna, ele 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 da faixa ou incorreto, defina um comportamento relevante. Caso contrário, exiba as informações de contato, um campo por linha.

- **SAÍDA**

• O programa fecha e os contatos são perdidos para sempre!

- **Qualquer outra entrada é ignorada.**

Após a execução correta de um comando, o programa aguarda outro. Ele para quando o usuário digita EXIT.


Dê um nome relevante ao seu executável.



<http://www.cplusplus.com/reference/string/string/> e claro <http://www.cplusplus.com/reference/iomanip/>

Capítulo V

Exercício 02: O Trabalho do Seu Sonhos

	Exercício: 02
O emprego dos seus sonhos	
Diretório de entrega: ex02/	
Arquivos para entrega: Makefile, Account.cpp, Account.hpp, tests.cpp Funções proibidas:	
Nenhuma	



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

Hoje é seu primeiro dia na *GlobalBanksters United*. Depois de passar com sucesso nos testes de recrutamento (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 impressionado com a rapidez com que você instalou o *Adobe Reader*. Esse pequeno extra fez toda a diferença e ajudou você a derrotar todos os seus oponentes (também conhecidos como os outros candidatos): você conseguiu!

De qualquer forma, seu gerente acabou de lhe dar um trabalho para fazer. Sua primeira tarefa é recriar um arquivo perdido. Algo deu errado e um arquivo de origem foi excluído por engano. Infelizmente, seus colegas não sabem o que é Git e usam pen drives para compartilhar código. Nesse ponto, faria sentido sair daqui agora mesmo. No entanto, você decide ficar. Desafio aceito!

Seus colegas desenvolvedores lhe deram um monte de arquivos. Compilar tests.cpp revela que o arquivo ausente 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 em C++ puro e incrível. Após algumas compilações malsucedidas, seu programa passa nos testes. Sua saída corresponde perfeitamente à salva no arquivo de log (**exceto pelos carimbos de data/hora**, que obviamente serão diferentes, já que os testes salvos no arquivo de log foram executados antes de você ser contratado).

Nossa, você é impressionante!



A ordem em que os destrutores são chamados pode variar dependendo do seu compilador/sistema operacional. Portanto, seus destrutores podem ser chamados em uma ordem inversa.



A conclusão do exercício 02 não é obrigatória para aprovação neste módulo.

Capítulo VI

Submissão e avaliação por pares

Envie sua tarefa para o seu repositório Git como de costume. Somente o trabalho dentro do seu repositório será avaliado durante a defesa. Não hesite em verificar os nomes dos seus arquivos para garantir que estejam corretos.



??????????? XXXXXXXXXXXX = \$3\$15bc138aca1e76ec6f4cfd0797ec037