



C++ - Módulo 00

Namespaces, classes, funções de membro, streams stdio,
listas de inicialização, static, const e algumas outras coisas
básicas

Resumo:

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

Versão: 8

Conteúdo

I	Introdução	2
II	Regras gerais	3
III	Exercício 00: Megafone	5
IV	Exercício 01: Minha Agenda Incrível	6
V	Exercício 02: O emprego dos seus sonhos	8

Capítulo I

Introdução

C++ é uma linguagem de programação de uso geral criada por Bjarne Stroustrup como uma extensão da linguagem de programação C, ou "C with Classes" (fonte: [Wikipedia](#)).

O objetivo desses módulos é apresentá-lo à **Programação Orientada a Objetos**.

Este será o ponto de partida de sua jornada C++. Muitos idiomas são recomendados para aprender OOP. Decidimos escolher C++, pois é derivado de seu velho amigo C.

Por se tratar de uma linguagem complexa e para manter as coisas simples, seu código estará em conformidade com o padrão C++98.

Sabemos que o C++ moderno é muito diferente em muitos aspectos. Portanto, se você deseja se tornar um desenvolvedor C++ proficiente, cabe a você ir além do 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 de exercícios serão nomeados desta forma: `ex00`, `ex01`, ... , `exn`
- Nomeie seus arquivos, classes, funções, funções de membro e atributos conforme exigido em As diretrizes.
- Escreva os nomes das classes no formato **UpperCamelCase**. Arquivos contendo código de classe serão sempre ser nomeado de acordo com o nome da classe. Por exemplo:
`ClassName.hpp/ClassName.h`, `ClassName.cpp` ou `ClassName.tpp`. Então, se você tiver um arquivo de cabeçalho contendo a definição de uma classe "BrickWall" que representa uma parede de tijolos, seu nome será `BrickWall.hpp`.
- A menos que especificado de outra forma, todas as mensagens de saída devem ser encerradas com uma nova linha caractere e exibido na saída padrão.
- *Adeus Norminette!* Nenhum estilo de codificação é aplicado nos módulos C++. Você pode seguir o seu favorito. Mas lembre-se de que um código que seus pares 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 codificando em C. Hora de C++! Portanto:

- Você tem permissão para usar quase tudo da biblioteca padrão. Portanto, em vez de se ater ao que você já sabe, seria inteligente usar o máximo possível as versões em C++ das funções C às quais você 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: `*printf()`, `*alloc()` e `free()`. Se você usá-los, sua nota será 0 e pronto.

- Observe que, a menos que explicitamente declarado de outra forma, o namespace using <ns_name> e palavras-chave de amigos são proibidas. Caso contrário, sua nota será -42.
- **Você tem permissão para usar o STL apenas no Módulo 08 e 09.** Isso significa: sem **contêineres** (vetor/lista/mapa/e assim por diante) e sem **algoritmos** (qualquer coisa que requeira incluir o cabeçalho <algorithm>) até então. Caso contrário, sua nota será -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 09, suas aulas devem ser elaboradas no **Ortodoxo 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 ser capaz de usar cada um de seus cabeçalhos independentemente dos outros. Assim, eles devem incluir todas as dependências de que precisam. No entanto, você deve evitar o problema de inclusão dupla adicionando **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 essas atribuições não são verificadas por um programa, sinta-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! Realmente, faça isso.
- Por Odin, por Thor! Use seu cérebro!!!




Você terá que implementar muitas classes. Isso pode parecer tedioso, a menos que você seja capaz de criar o script de 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 poderia perder muita informação útil! Não hesite em ler sobre conceitos teóricos.

Capítulo III

Exercício 00: Megafone

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

Apenas para ter certeza de que todos estão acordados, escreva um programa que se comporte da seguinte maneira:


```
$>./megafone "shhhhh... acho que os alunos estão dormindo..."
SHHHHH... ACHO QUE OS ALUNOS ESTÃO DORMINDO...
! "Desculpe alunos, eu pensei que esta coisa estava desligada."
$>./megafone Droga Droga
Droga! DESCULPE ALUNOS, PENSEI QUE ESTA COISA ESTAVA DESLIGADA. $>./
megafone
* RUÍDO DE FEEDBACK ALTO E INSUPORTÁVEL * $>
```



Resolva os exercícios em linguagem C++.

Capítulo IV

Exercício 01: Meu Incrível Lista telefônica

	Exercício: 01
Minha agenda incrível	
Diretório de entrega: ex01/	
Arquivos a serem entregues: 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 incrível de baixa qualidade.

Você tem que implementar duas classes:

- **Agenda**

- Possui uma matriz de contatos.
- Pode armazenar no máximo **8 contatos**. Se o usuário tentar adicionar um 9º contato, substitua o mais antigo pelo novo.
- Observe que a alocação dinâmica é proibida.

- **Contato**

- Representa um contato da agenda.

Em seu código, a lista telefônica deve ser instanciada como uma instância da classe **PhoneBook**. Mesma coisa para os contatos. Cada um deles deve ser instanciado como uma instância da classe **Contact**. Você é livre para projetar 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 agenda telefônica está vazia e o usuário é solicitado a inserir um de três comandos. O programa só aceita ADICIONAR, PESQUISAR e SAIR.

- **ADICIONAR:** salva um novo contato

- Se o usuário inserir esse comando, ele será solicitado a inserir as informações do novo contato, um campo por vez. Depois de preencher 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.

- **PESQUISAR:** exibe um contato específico

- Exiba 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 ('|') os separa. O texto deve estar 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 do intervalo ou errado, defina um comportamento relevante. Caso contrário, exiba as informações de contato, um campo por linha.

- **SAIR**

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

- **Qualquer outra entrada é descartada.**

Depois que um comando foi executado corretamente, o programa espera por outro. Ele para quando o usuário insere 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 Seu Trabalho sonhos

	Exercício: 02
O emprego dos seus sonhos	
Diretório de entrega: ex02/	
Arquivos a serem entregues: 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 da intranet do módulo.

Hoje é seu primeiro dia no *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 surpreso com a rapidez com que você instalou o *Adobe Reader*. Aquele pequeno extra fez toda a diferença e ajudou você a derrotar todos os seus adversários (também conhecidos como os outros candidatos): você conseguiu!

De qualquer forma, seu gerente apenas lhe deu algum 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 chaves USB para compartilhar código. Neste ponto, faria sentido deixar este lugar agora. No entanto, você decide ficar. Desafio aceito!

Seus colegas desenvolvedores fornecem vários arquivos. A compilação de tests.cpp revela que o arquivo ausente é Account.cpp. Para sua 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 incrível. Depois de algumas compilações malsucedidas, seu programa passa nos testes. Sua saída corresponde perfeitamente àquela salva no arquivo de log (**exceto para os timestamps** que obviamente serão diferentes, pois os testes salvos no arquivo de log foram executados antes de você ser contratado).

Porra, você é impressionante!



A ordem na qual os destruidores são chamados pode diferir dependendo do seu compilador/sistema operacional. Então seus destruidores podem ser chamados uma ordem inversa.



Você pode passar neste módulo sem fazer o exercício 02.