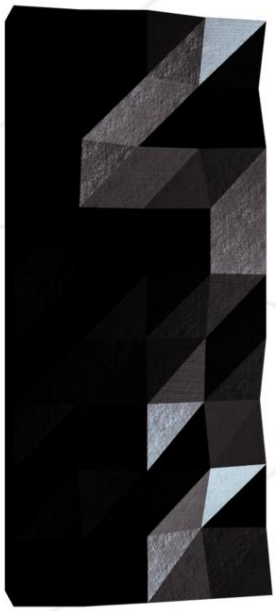

# C++ - Módulo 00

Namespaces, 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*

# Conteúdo

# 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 com Classes" (fonte: Wikipedia).*

O objetivo desses módulos é apresentar **a Programação Orientada a Objetos.**
Este será o ponto de partida de sua jornada em C++. Muitas linguagens são recomendadas para aprender OOP. Decidimos escolher C++ porque ele é derivado do 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.

Estamos cientes de 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 após o 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 deverá ser compilado se você adicionar o sinalizador -std=c++98

### Convenções de formatação e nomenclatura

- Os diretórios dos exercícios serão nomeados desta forma: ex00, ex01, ... , ex
- Nomeie seus arquivos, classes, funções, funções de membro e atributos conforme exigido em As diretrizes.
- Escreva nomes de 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 finalizadas com uma nova linha caractere e exibido na saída padrão.
- *Adeus Norminette!* Nenhum estilo de codificação é imposto nos módulos C++. Você pode seguir o seu favorito. Mas tenha em mente 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. É hora de C++! Portanto:

- Você tem permissão para usar quase tudo da biblioteca padrão. Assim, em vez de se ater ao que você já sabe, seria inteligente usar o máximo possível as versões C++ das funções C com as quais você está acostumado.
- Entretanto, você não pode usar nenhuma outra biblioteca externa. Isso significa que C++ 11 (e formas derivadas) e bibliotecas 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, salvo indicação explícita em contrário, o namespace using <ns_name> e palavras-chave de amigos são proibidas. Caso contrário, sua nota será -42.
- **É permitido utilizar o STL somente nos Módulos 08 e 09.** Isso significa: nenhum **contêiner** (vetor/lista/mapa/e assim por diante) e nenhum **algoritmo** (qualquer coisa que exija a inclusão do cabeçalho <algorithm>) até então. Caso contrário, sua nota será -42.

**Alguns requisitos de design**

- 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 planejadas no **estilo Ortodoxo Forma Canônica, exceto quando explicitamente indicado de outra forma.**
- Qualquer implementação de função colocada em um arquivo de cabeçalho (exceto modelos de função) significa 0 para o exercício.
- Você deve ser capaz de usar cada um dos seus cabeçalhos independentemente dos outros. Assim, eles devem incluir todas as dependências de que necessitam. No entanto, você deve evitar o problema da 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, fique à vontade para fazê-lo, desde que entregue os arquivos obrigatórios.
- Às vezes, as orientações 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 consiga criar um script em seu editor de texto favorito.

Você tem uma certa liberdade para completar os exercícios. Porém, siga as regras obrigatórias e não seja preguiçoso. Você poderia 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 | |

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

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

Resolva os exercícios em C++.

# Capítulo IV

# Exercício 01: Meu Incrível Lista telefônica

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

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

Você tem que implementar duas classes:

• **Lista telefônica**

ÿ Possui uma variedade de contatos.

ÿ Pode armazenar no máximo **8 contatos.** Caso o usuário tente adicionar um 9º contato, substitua o mais antigo pelo novo.

ÿ Observe que a alocação dinâmica é proibida.

• **Contato**

ÿ Significa um contato da agenda telefônica.

No 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 **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 que 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 está vazia e o usuário é solicitado a inserir uma de três comandos. O programa aceita apenas ADD, SEARCH e EXIT.

- **ADICIONAR:** salve um novo contato
  - ÿ Se o usuário inserir esse comando, ele será solicitado a inserir as informações do novo contato, um campo por vez. Depois que todos os campos forem preenchidos, adicione o contato à lista telefônica.
  - ÿ Os campos de contato são: nome, sobrenome, apelido, número de telefone e segredo mais obscuro. Um contato salvo não pode ter campos vazios.
- **PESQUISA:** exibe um contato específico
  - ÿ Exiba os contatos salvos como uma lista de **4 colunas:** índice, nome, sobrenome nome e apelido.
  - ÿ Cada coluna deve ter **10 caracteres** de largura. Um caractere de barra vertical ('|') os separa. O texto deve estar alinhado à direita. Se o texto for maior que a coluna, ele deverá ser truncado e o último caractere exibível deverá ser substituído por um ponto ('.').
  - ÿ Em seguida, solicite novamente ao usuário o índice da entrada a ser exibido. Se o índice estiver fora do intervalo ou errado, defina um comportamento relevante. Caso contrário, exiba as informações de contato, um campo por linha.
- **SAÍDA**
  - ÿ O programa é encerrado e os contatos são perdidos para sempre!
- **Qualquer outra entrada é descartada.**

Depois que um comando for executado corretamente, 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 trabalho dos seus sonhos | |
| Diretório de entrega: ex02/ | |
| Arquivos para entrega: Makefile, Account.cpp, Account.hpp, testes.cpp Funções proibidas: | |
| Nenhuma | |

Account.hpp, testes.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.* 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 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 testes.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 com falha, seu programa passa nos testes. Sua saída corresponde perfeitamente àquela salva no arquivo de log **(exceto pelos carimbos de data e hora** que obviamente serão diferentes, pois os testes salvos no arquivo de log foram executados antes de você ser contratado).

Droga, você é impressionante!

A ordem em que os destruidores são chamados pode variar 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.

# Capítulo VI

# Envio e avaliação por pares

Entregue sua tarefa em seu repositório Git normalmente. Somente o trabalho dentro do seu repositório será avaliado durante a defesa. Não hesite em verificar os nomes dos seus arquivos para garantir que estão corretos.

?????????? XXXXXXXXXX = $3$$f15bc138aca1e76ec6f4cfd0797ec037