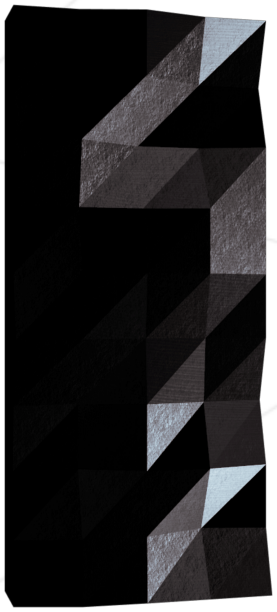

C++ - Módulo 00

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

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

Versão: 11.0

Sumário

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 introduzi-lo à **Programação Orientada a Objetos**. Este será o ponto de partida da sua jornada em C++. Muitas linguagens são recomendadas para aprender OOP. Decidimos escolher C++ já que ela deriva da sua velha amiga C. Já que C++ é uma linguagem complexa, e para manter as coisas simples, seu código seguirá o padrão C++98.

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

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 flags `-Wall -Wextra -Werror`
- Seu código ainda deve compilar se você adicionar a flag `-std=c++98`

Formatação e convenções de nomenclatura

- Os diretórios dos exercícios serão nomeados desta 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**. Arquivos contendo código de classe sempre serão nomeados de acordo com o nome da classe. Por exemplo:
  `NomeDaClasse.hpp/NomeDaClasse.h`, `NomeDaClasse.cpp`, ou `NomeDaClasse.tpp`.
  Então, se você tiver um arquivo de cabeçalho contendo a definição de uma classe "ParedeDeTijolo"representando uma parede de tijolos, seu nome será `ParedeDeTijolo.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 que o código que seus 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 programando em C. É hora de C++! Portanto:

- Você pode usar quase tudo da biblioteca padrão. Assim, em vez de se ater ao que você já conhece, seria inteligente usar as versões em C++ das funções C que você está acostumado o máximo possível.

- No entanto, você não pode usar nenhuma outra biblioteca externa. Isso significa que 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.

- Observe que, a menos que explicitamente indicado de outra forma, as palavras-chave `using namespace <ns_name>` e `friend` são proibidas. Caso contrário, sua nota será -42.

- **Você só pode usar a STL nos Módulos 08 e 09.** Isso significa: nenhum **Contêiner** (vector/list/map, e assim por diante) e nenhum **Algoritmo** (qualquer coisa que exija a inclusão do cabeçalho `<algorithm>`) até lá. Caso contrário, sua nota será -42.

Alguns requisitos de design

- Vazamento de memória ocorre em C++ também. Quando você aloca memória (usando a palavra-chave `new`), você deve evitar **vazamentos de memória**.

- Do Módulo 02 ao Módulo 09, suas classes devem ser projetadas 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 de seus cabeçalhos independentemente de outros. Portanto, eles devem incluir todas as dependências de que precisam. 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 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 você 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 (consulte o capítulo Norma sobre o Makefile).`

Você terá que implementar muitas classes. Isso pode parecer tedioso, a menos que você seja capaz de criar scripts para 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 perderia muitas informações úteis! Não hesite em ler sobre conceitos teóricos.

Capítulo III

Instruções de IA

Contexto

Este projeto foi desenvolvido para ajudá-lo a descobrir os blocos de construção fundamentais do seu treinamento em TIC.

Para ancorar adequadamente os conhecimentos e habilidades-chave, é essencial adotar uma abordagem criteriosa ao uso de ferramentas e suporte de IA.

A verdadeira aprendizagem fundamental exige um esforço intelectual genuíno — através de desafios, repetição e trocas de aprendizagem entre pares.

Para uma visão geral mais completa de nossa posição sobre a IA — como ferramenta de aprendizagem, como parte do currículo de TIC e como expectativa no mercado de trabalho — consulte as perguntas frequentes dedicadas na intranet.

Mensagem principal

- Construa bases sólidas sem atalhos.
- Desenvolva verdadeiramente habilidades técnicas e de poder.
- Experimente a verdadeira aprendizagem entre pares, comece a aprender como aprender e resolver novos problemas.
- A jornada de aprendizagem é mais importante que o resultado.
- Aprenda sobre os riscos associados à IA e desenvolva práticas eficazes de controle e contramedidas para evitar armadilhas comuns.

Regras para o aluno:

- Você deve aplicar o raciocínio às suas tarefas atribuídas, especialmente antes de recorrer à IA.
- Você não deve pedir respostas diretas à IA.
- Você deve aprender sobre a abordagem global da 42 em relação à IA.

Resultados da fase:

Nesta fase fundamental, você obterá os seguintes resultados:

- Obter bases sólidas em tecnologia e codificação.
- Saber por que e como a IA pode ser perigosa durante esta fase.

Comentários e exemplo:

- Sim, sabemos que a IA existe — e sim, ela pode resolver seus projetos. Mas você está aqui para aprender, não para provar que a IA aprendeu. Não perca seu tempo (nem o nosso) apenas para demonstrar que a IA pode resolver o problema dado.
- Aprender na 42 não é sobre saber a resposta — é sobre desenvolver a capacidade de encontrar uma. A IA lhe dá a resposta diretamente, mas isso o impede de construir seu próprio raciocínio. E o raciocínio leva tempo, esforço e envolve falhas. O caminho para o sucesso não deve ser fácil.
- Lembre-se de que durante os exames, a IA não estará disponível — sem internet, sem smartphones, etc. Você perceberá rapidamente se confiou demais na IA em seu processo de aprendizagem.
- A aprendizagem entre pares o expõe a diferentes ideias e abordagens, melhorando suas habilidades interpessoais e sua capacidade de pensar de forma divergente. Isso é muito mais valioso do que apenas conversar com um bot. Então não seja tímido — converse, faça perguntas e aprenda juntos!
- Sim, a IA fará parte do currículo — tanto como ferramenta de aprendizagem quanto como um tópico em si. Você até terá a chance de construir seu próprio software de IA. Para saber mais sobre nossa abordagem crescente, consulte a documentação disponível na intranet.

✓ Boa prática:

> Estou travado em um novo conceito. Pergunto a alguém próximo como ele abordou isso. Conversamos por 10 minutos — e de repente, clica. Entendi.

✗ Má prática:

> Uso secretamente a IA, copio algum código que parece certo. Durante a avaliação entre pares, não consigo explicar nada. Eu falho. Durante o exame — sem IA — estou travado novamente. Eu falho.

Capítulo IV

Exercício 00: Megaphone

	Exercice : 00
Megaphone	
Pasta de entrega : *ex*00/	
Arquivos para entregar : `Makefile, megaphone.cpp`	
Funções não permitidas : `Nenhuma`	

Apenas para garantir que todos estejam acordados, escreva um programa que produza a seguinte saída:

```
$>./megaphone "shhhhh... I think the students are asleep..."
SHHHHH... I THINK THE STUDENTS ARE ASLEEP...
$>./megaphone Damnit " ! " "Sorry students, I thought this thing was off."
DAMNIT ! SORRY STUDENTS, I THOUGHT THIS THING WAS OFF.
$>./megaphone
* LOUD AND UNBEARABLE FEEDBACK NOISE *
$>
```

Resolva os exercícios de maneira C++.

Capítulo V

Exercício 01: Minha Incrível Agenda Telefônica

	Exercice : 01
Minha Incrível Agenda Telefônica	
Pasta de entrega : $ex01/$	
Arquivos para entregar : `Makefile, *.cpp, *.{h, hpp}`	
Funções não permitidas : `Nenhuma`	

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

Você tem que implementar duas classes:

- **PhoneBook**
  - Ela tem um array de contatos.
  - Ela 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 alocação dinâmica é proibida.
- **Contact**
  - Representa um contato da agenda telefônica.

No seu código, a agenda 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 projetar as classes como quiser, mas tenha em mente que qualquer coisa que sempre será usada dentro de uma classe é privada, e que qualquer coisa que pode ser usada fora de uma classe é pública.

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 ADD, SEARCH e EXIT.

- **ADD**: salvar um novo contato
  - Se o usuário entrar com este comando, ele é solicitado a inserir as informações do novo contato um campo por vez. Uma vez que todos os campos tenham sido completados, adicione o contato à agenda telefônica.
  - Os campos do contato são: primeiro nome, sobrenome, apelido, número de telefone e segredo mais obscuro. Um contato salvo não pode ter campos vazios.
- **SEARCH**: exibir um contato específico
  - Exiba 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 pipe ('|') os 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 ('.').
  - Então, solicite ao usuário novamente 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.
- **EXIT**
  - O programa fecha e os contatos são perdidos para sempre!
- **Qualquer outra entrada é ignorada.**

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

Dê um nome relevante ao seu executável.

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

Capítulo VI

Exercício 02: O Emprego Dos Seus Sonhos

	Exercice : 02
O Emprego Dos Seus Sonhos	
Pasta de entrega : $ex02/$	
Arquivos para entregar : `Makefile, Account.cpp, Account.hpp, tests.cpp`	
Funções não permitidas : `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 na *GlobalBanksters United*. Depois de passar com sucesso nos testes de recrutamento (graças a alguns truques do *Microsoft Office* que um amigo te mostrou), você entrou para a equipe de desenvolvimento. Você também sabe que o recrutador ficou impressionado com a rapidez com que você instalou o *Adobe Reader*. Aquele pequeno extra fez toda a diferença e te ajudou a derrotar todos os seus oponentes (também conhecidos como os outros candidatos): você conseguiu!

De qualquer forma, seu gerente acabou de te dar um trabalho para fazer. Sua primeira tarefa é reconstruir um arquivo perdido. Algo deu errado e um arquivo fonte foi excluído 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. No entanto, você decide ficar. Desafio aceito!

Seus colegas desenvolvedores te dão um monte de arquivos. Compilar `tests.cpp` revela que o arquivo ausente é `Account.cpp`. Felizmente para você, 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. Após algumas compilações falhadas, seu programa passa nos testes. Sua saída corresponde perfeitamente àquela salva no arquivo de log (**exceto pelos timestamps** que obviamente diferirão, já que os testes salvos no arquivo de log foram executados antes de você ser contratado).

Droga, você é impressionante!

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

Completar o exercício 02 não é obrigatório para passar neste módulo.

Capítulo VII

Entrega e avaliação por pares

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

Durante a avaliação, uma breve **modificação do projeto** pode ser ocasionalmente solicitada. Isso pode envolver uma pequena mudança de comportamento, algumas linhas de código para escrever ou reescrever, ou um recurso fácil de adicionar.

Embora esta etapa possa **não ser aplicável a todos os projetos**, você deve estar preparado para ela se for mencionada nas diretrizes de avaliação.

Esta etapa visa verificar sua compreensão real de uma parte específica do projeto. A modificação pode ser realizada em qualquer ambiente de desenvolvimento que você escolher (por exemplo, sua configuração usual), e deve ser viável em poucos minutos — a menos que um prazo específico seja definido como parte da avaliação.
Você pode, por exemplo, ser solicitado a fazer uma pequena atualização em uma função ou script, modificar uma exibição ou ajustar uma estrutura de dados para armazenar novas informações, etc.

Os detalhes (escopo, alvo, etc.) serão especificados nas **diretrizes de avaliação** e podem variar de uma avaliação para outra para o mesmo projeto.

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