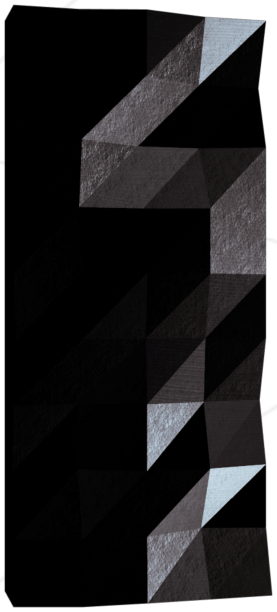

C++ - Módulo 07

Templates C++

Preâmbulo:
Este documento contém os exercícios do Módulo 07 dos módulos de C++.

Versão: 10.0

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 apresentá-lo à **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++ já que é derivado do seu velho amigo C. Como esta é 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 bem diferente em muitos aspectos. Então, se você quiser se tornar um desenvolvedor C++ proficiente, cabe a você ir mais longe após o 42 Common Core!

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: Comece com algumas funções

	Exercice : 00
Comece com algumas funções	
Pasta de entrega : $ex00/$	
Arquivos para entregar : `Makefile, main.cpp, whatever.{h, hpp}`	
Funções não permitidas : `Nenhuma`	

Implemente os seguintes templates de função:

- `swap`: Troca os valores de dois parâmetros fornecidos. Não retorna nada.
- `min`: Compara os dois valores passados como parâmetros e retorna o menor. Se forem iguais, retorna o segundo.
- `max`: Compara os dois valores passados como parâmetros e retorna o maior. Se forem iguais, retorna o segundo.

Essas funções podem ser chamadas com qualquer tipo de argumento. O único requisito é que os dois argumentos devem ter o mesmo tipo e devem suportar todos os operadores de comparação.

`Os templates devem ser definidos nos arquivos de header.`

Executando o seguinte código:

```
int     main( void ) {

    int a = 2;
    int b = 3;

    ::swap( a, b );
    std::cout << "a = " << a << ", b = " << b << std::endl;
    std::cout << "min( a, b ) = " << ::min( a, b ) << std::endl;
    std::cout << "max( a, b ) = " << ::max( a, b ) << std::endl;

    std::string c = "chaine1";
    std::string d = "chaine2";

    ::swap(c, d);
    std::cout << "c = " << c << ", d = " << d << std::endl;
    std::cout << "min( c, d ) = " << ::min( c, d ) << std::endl;
    std::cout << "max( c, d ) = " << ::max( c, d ) << std::endl;

    return 0;
}
```

Deve resultar em:

```
a = 3, b = 2
min(a, b) = 2
max(a, b) = 3
c = chaine2, d = chaine1
min(c, d) = chaine1
max(c, d) = chaine2
```

Capítulo V

Exercício 01: Iter

	Exercice : 01
Iter	
Pasta de entrega : $ex01/$	
Arquivos para entregar : `Makefile, main.cpp, iter.{h, hpp}`	
Funções não permitidas : `Nenhuma`	

Implemente um template de função `iter` que recebe 3 parâmetros e não retorna nada.

- O primeiro parâmetro é o endereço de um array.
- O segundo é o comprimento do array, passado como um valor `const`.
- O terceiro é uma função que será chamada em cada elemento do array.

Entregue um arquivo `main.cpp` que contenha seus testes. Forneça código suficiente para gerar um executável de teste.

Seu template de função `iter` deve funcionar com qualquer tipo de array. O terceiro parâmetro pode ser um template de função instanciado.

A função passada como terceiro parâmetro pode receber seu argumento por `const` referência ou referência não-`const`, dependendo do contexto.

`Pense cuidadosamente sobre como suportar elementos const e não-const em sua função iter.`

Capítulo VI

Exercício 02: Array

	Exercice : 02
Array	
Pasta de entrega : $ex02/$	
Arquivos para entregar : `Makefile, main.cpp, Array.{h, hpp}` e `arquivo opcional: Array.tpp`	
Funções não permitidas : `Nenhuma`	

Desenvolva um template de classe **Array** que contenha elementos do tipo `T` e que implemente o seguinte comportamento e funções:

- Construção sem parâmetro: Cria um array vazio.
- Construção com um unsigned int `n` como parâmetro: Cria um array de `n` elementos inicializados por padrão.
  Dica: Tente compilar `int * a = new int();` e então exibir `*a`.
- Construção por cópia e operador de atribuição. Em ambos os casos, modificar o array original ou sua cópia após a cópia não deve afetar o outro array.
- Você DEVE usar o operador `new[]` para alocar memória. Alocação preventiva (alocar memória com antecedência) é proibida. Seu programa nunca deve acessar memória não alocada.
- Os elementos podem ser acessados através do operador de subscrito: [ ].
- Ao acessar um elemento com o operador [ ], se o seu índice estiver fora dos limites, uma `std::exception` é lançada.
- Uma função membro `size()` que retorna o número de elementos no array. Esta função membro não recebe parâmetros e não deve modificar a instância atual.

Como de costume, certifique-se de que tudo funciona como esperado e entregue um arquivo `main.cpp` que contenha seus testes.

Capítulo VII

Entrega e avaliação por pares

Entregue seu projeto em seu repositório `Git` como de costume. Apenas o trabalho dentro do seu repositório será avaliado durante a defesa. Não hesite em verificar os nomes de suas pastas e 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.

16D85ACC441674FBA2DF65190663F43A243E8FA5424E49143B520D3DF8AF68036E47
114F20A16827E1B16612137E59ECD492E468BC6CD109F65388DC57A58E8942585C8
D193B96732206