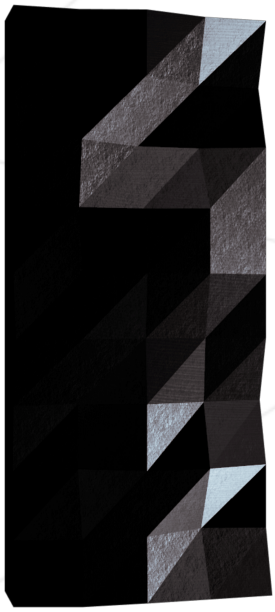

C++ - Módulo 08

Containers Templateados, Iteradores, Algoritmos

Preâmbulo:
Este documento contém os exercícios do Módulo 08 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 apresentar a você a **Programação Orientada a Objetos**. Este será o ponto de partida de sua jornada em C++. Muitas linguagens são recomendadas para aprender POO. Decidimos escolher C++ já que é derivado de seu velho amigo C. Por ser 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

Regras Específicas do Módulo

Você notará que, neste módulo, os exercícios podem ser resolvidos SEM os Containers padrão e SEM os Algoritmos padrão.

No entanto, **usá-los é precisamente o objetivo deste Módulo**.
Você deve usar a STL — especialmente os **Containers** (vector/list/map/e assim por diante) e os **Algoritmos** (definidos no cabeçalho `<algorithm>`) — sempre que apropriado.
Além disso, você deve usá-los o máximo que puder.
Assim, faça o seu melhor para aplicá-los onde for apropriado.

Você receberá uma nota muito ruim se não o fizer, mesmo que seu código funcione como esperado. Por favor, não seja preguiçoso.

Você pode definir seus templates nos arquivos de header como de costume. Ou, se você quiser você pode escrever suas declarações de template nos arquivos de header e escrever suas implementações em arquivos .tpp. Em qualquer caso, os arquivos de header são obrigatórios enquanto os arquivos .tpp são opcionais.

Capítulo IV

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 V

Exercício 00: Easy find

	Exercice : 00
Easy find	
Pasta de entrega : $ex00/$	
Arquivos para entregar : `Makefile, main.cpp, easyfind.{h, hpp}` e `arquivo opcional: easyfind.tpp`	
Funções não permitidas : `Nenhuma`	

Um primeiro exercício fácil é a maneira de começar com o pé direito.

Escreva um template de função `easyfind` que aceite um tipo `T`. Ele recebe dois parâmetros: o primeiro é do tipo `T`, e o segundo é um inteiro.

Assumindo que `T` é um container **de inteiros**, esta função tem que encontrar a primeira ocorrência do segundo parâmetro no primeiro parâmetro.

Se nenhuma ocorrência for encontrada, você pode lançar uma exceção ou retornar um valor de erro de sua escolha. Se você precisa de alguma inspiração, analise como os containers padrão se comportam.

Claro, implemente e entregue seus próprios testes para garantir que tudo funcione como esperado.

`Você não precisa lidar com containers associativos.`

Capítulo VI

Exercício 01: Span

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

Desenvolva uma classe **Span** que possa armazenar um máximo de `N` inteiros. `N` é uma variável unsigned int e será o único parâmetro passado para o construtor.

Esta classe terá uma função membro chamada `addNumber()` para adicionar um único número ao Span. Ele será usado para preenchê-lo. Qualquer tentativa de adicionar um novo elemento se já houver `N` elementos armazenados deve lançar uma exceção.

Em seguida, implemente duas funções membro: `shortestSpan()` e `longestSpan()`

Eles irão, respectivamente, descobrir o menor span ou o maior span (ou distância, se preferir) entre todos os números armazenados e retorná-lo. Se não houver números armazenados, ou apenas um, nenhum span pode ser encontrado. Assim, lance uma exceção.

Claro, você escreverá seus próprios testes e eles serão muito mais completos do que os abaixo. Teste seu `Span` com pelo menos 10.000 números. Mais seria ainda melhor.

Executando este código:

```
int main()
{
  Span sp = Span(5);

  sp.addNumber(6);
  sp.addNumber(3);
  sp.addNumber(17);
  sp.addNumber(9);
  sp.addNumber(11);

  std::cout << sp.shortestSpan() << std::endl;
  std::cout << sp.longestSpan() << std::endl;

  return 0;
}
```

Deve mostrar:

```
$> ./ex01
2
14
$>
```

Por último, mas não menos importante, seria maravilhoso preencher seu `Span` usando um **intervalo de iteradores**. Fazer milhares de chamadas para `addNumber()` é tão irritante. Implemente uma função membro para adicionar vários números ao seu `Span` em uma única chamada.

Se você não tem ideia, estude os Containers. Algumas funções membro recebem um intervalo de iteradores para adicionar uma sequência de elementos ao container.

Capítulo VII

Exercício 02: Abominação Mutada

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

Agora, é hora de passar para coisas mais sérias. Vamos desenvolver algo estranho.

O container `std::stack` é muito bom. Infelizmente, é um dos únicos Containers STL que NÃO são iteráveis. Isso é muito ruim.

Mas por que aceitaríamos isso? Especialmente se pudermos nos dar ao luxo de massacrar a pilha original para criar os recursos que faltam.

Para reparar esta injustiça, você deve tornar o container `std::stack` iterável.

Escreva uma classe **MutantStack**. Ela **será implementada em termos de** uma `std::stack`. Ela oferecerá todas as suas funções membro, além de uma recurso adicional: **iteradores**.

Claro, você escreverá e entregará seus próprios testes para garantir que tudo funcione como esperado.

Encontre um exemplo de teste abaixo.

```
int main()
{
MutantStack<int>    mstack;

mstack.push(5);
mstack.push(17);

std::cout << mstack.top() << std::endl;

mstack.pop();

std::cout << mstack.size() << std::endl;

mstack.push(3);
mstack.push(5);
mstack.push(737);
//[...]
mstack.push(0);

MutantStack<int>::iterator it = mstack.begin();
MutantStack<int>::iterator ite = mstack.end();

++it;
--it;
while (it != ite)
{
    std::cout << *it << std::endl;
    ++it;
}
std::stack<int> s(mstack);
return 0;
}
```

Se você executá-lo uma primeira vez com seu MutantStack e uma segunda vez substituindo o MutantStack com, por exemplo, uma `std::list`, as duas saídas devem ser as mesmas. Claro, ao testar outro container, atualize o código abaixo com as funções membro correspondentes (`push()` pode se tornar `push_back()`).

Capítulo VIII

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 novamente os nomes de suas pastas e arquivos para garantir que eles estão 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.

16D85ACC441674FBA2DF65195A38EE36793A89EB04594B15725A1128E7E97B0E7B47
111668BD6823E2F873124B7E59B5CE94B47AB764CF0AB316999C56E5989B4B4F00C
91B619C70263F