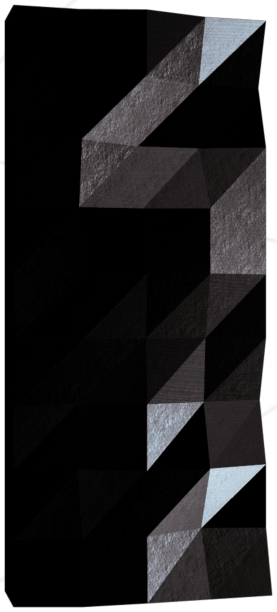

# C++ - Módulo 09

## STL

*Preâmbulo:*
*Este documento contém os exercícios do Módulo 09 dos módulos de C++.*

*Versão: 3.0*

# Sumário


| | | |
|---|---|---|
| **I** | **Introdução** | **2** |
| **II** | **Regras gerais** | **3** |
| **III** | **Regras específicas do módulo** | **6** |
| **IV** | **Instruções de IA** | **7** |
| **V** | **Exercício 00: Bitcoin Exchange** | **10** |
| **VI** | **Exercício 01: Reverse Polish Notation (Notação Polonesa Reversa)** | **12** |
| **VII** | **Exercício 02: PmergeMe** | **14** |
| **VIII** | **Entrega e avaliação por pares** | **17** |

# 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 é derivado do seu velho amigo C. Como esta é uma linguagem complexa, e para manter as coisas simples, o seu código deverá estar em conformidade com 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 além 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

É obrigatório usar os contêineres padrão para realizar cada exercício neste módulo.

Uma vez que um contêiner seja usado, você não pode mais usá-lo para o resto do módulo.

`É aconselhável ler o subject por completo antes de fazer os exercícios.`

`Você deve usar pelo menos um contêiner para cada exercício, com exceção do exercício 02, que exige o uso de dois contêineres.`

Você deve enviar um `Makefile` para cada programa que compilará seus arquivos de origem para a saída necessária com as flags `-Wall`, `-Wextra` e `-Werror`.

Você deve usar c++, e seu Makefile não deve dar relink.

Seu `Makefile` deve conter pelo menos as regras `$(NAME)`, `all`, `clean`, `fclean` e `re`.

# 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: Bitcoin Exchange

|  | Exercice : 00 |
|---|---|
| Bitcoin Exchange | |
| Pasta de entrega : *ex*00/ | |
| Arquivos para entregar : `Makefile, main.cpp, BitcoinExchange.{cpp, hpp}` | |
| Funções não permitidas : `Nenhuma` | |

Você tem que criar um programa que exiba o valor de uma certa quantidade de bitcoin em uma determinada data.

Este programa deve usar um banco de dados em formato csv que representará o preço do bitcoin ao longo do tempo. Este banco de dados é fornecido com este subject.

O programa receberá como entrada um segundo banco de dados, armazenando os diferentes preços/datas para avaliar.

Seu programa deve respeitar estas regras:

- O nome do programa é btc.
- Seu programa deve receber um arquivo como argumento.
- Cada linha neste arquivo deve usar o seguinte formato: "data | valor".
- Uma data válida sempre estará no seguinte formato: Ano-Mês-Dia.
- Um valor válido deve ser um float ou um inteiro positivo, entre 0 e 1000.

`Você deve usar pelo menos um contêiner em seu código para validar este exercício. Você deve lidar com possíveis erros com uma mensagem de erro apropriada.`

Aqui está um exemplo de um arquivo input.txt:

```
$> head input.txt
date | value
2011-01-03 | 3
2011-01-03 | 2
2011-01-03 | 1
2011-01-03 | 1.2
2011-01-09 | 1
2012-01-11 | -1
2001-42-42
2012-01-11 | 1
2012-01-11 | 2147483648
$>
```

Seu programa usará o valor em seu arquivo de entrada.

Seu programa deve exibir na saída padrão o resultado do valor multiplicado pela taxa de câmbio de acordo com a data indicada em seu banco de dados.

Se a data usada na entrada não existir em seu DB, então você deve usar a data mais próxima contida em seu DB. Tenha cuidado para usar a data inferior e não a superior.

O seguinte é um exemplo do uso do programa.

```
$> ./btc
Error: could not open file.
$> ./btc input.txt
2011-01-03 => 3 = 0.9
2011-01-03 => 2 = 0.6
2011-01-03 => 1 = 0.3
2011-01-03 => 1.2 = 0.36
2011-01-09 => 1 = 0.32
Error: not a positive number.
Error: bad input => 2001-42-42
2012-01-11 => 1 = 7.1
Error: too large a number.
$>
```

Aviso: O(s) contêiner(es) que você usar para validar este exercício não poderá(ão) mais ser usado(s) para o resto deste módulo.

# Capítulo VI

# Exercício 01: Reverse Polish Notation (Notação Polonesa Reversa)

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

Você deve criar um programa com estas restrições:

- O nome do programa é RPN.
- Seu programa deve receber uma expressão matemática polonesa reversa como um argumento.
- Os números usados nesta operação e passados como argumentos serão sempre menores que 10. O cálculo em si, mas também o resultado não levam em conta esta regra.
- Seu programa deve processar esta expressão e exibir o resultado correto na saída padrão.
- Se um erro ocorrer durante a execução do programa, uma mensagem de erro deve ser exibida no erro padrão.
- Seu programa deve ser capaz de lidar com operações com estes tokens: "+ - / *".

Você deve usar pelo menos um contêiner em seu código para validar este exercício.

Você não precisa lidar com parênteses ou números decimais.

Aqui está um exemplo de uso padrão:

```
$> ./RPN "8 9 * 9 - 9 - 9 - 4 - 1 +"
42
$> ./RPN "7 7 * 7 -"
42
$> ./RPN "1 2 * 2 / 2 * 2 4 - +"
0
$> ./RPN "(1 + 1)"
Error
$>
```

Aviso: O(s) contêiner(es) que você usou no exercício anterior estão proibidos aqui. O(s) contêiner(es) que você usou para validar este exercício não será(ão) utilizável(is) para o resto deste módulo.

# Capítulo VII

# Exercício 02: PmergeMe

|  | Exercice : 02 |
|---|---|
| PmergeMe | |
| Pasta de entrega : $ex02/$ | |
| Arquivos para entregar : `Makefile, main.cpp, PmergeMe.{cpp, hpp}` | |
| Funções não permitidas : `Nenhuma` | |

Você deve criar um programa com estas restrições:

- O nome do programa é PmergeMe.
- Seu programa deve ser capaz de usar uma sequência de inteiros positivos como um argumento.
- Seu programa deve usar o algoritmo de ordenação merge-insert para ordenar a sequência de inteiros positivos.

Para esclarecer, sim, você precisa usar o algoritmo de Ford-Johnson. (fonte: Art Of Computer Programming, Vol.3. Merge Insertion, Página 184.)

- Se um erro ocorrer durante a execução do programa, uma mensagem de erro deve ser exibida no erro padrão.

Você deve usar pelo menos dois contêineres diferentes em seu código para validar este exercício. Seu programa deve ser capaz de lidar com pelo menos 3000 inteiros diferentes.

É fortemente aconselhável implementar seu algoritmo para cada contêiner e, portanto, evitar usar uma função genérica.

Aqui estão algumas diretrizes adicionais sobre as informações que você deve exibir linha por linha na saída padrão:

- Na primeira linha você deve exibir um texto explícito seguido pela sequência de inteiros positivos não ordenada.
- Na segunda linha você deve exibir um texto explícito seguido pela sequência de inteiros positivos ordenada.
- Na terceira linha, você deve exibir uma mensagem explícita indicando o tempo gasto pelo seu algoritmo, especificando o primeiro contêiner usado para ordenar a sequência de inteiros positivos.
- Na última linha você deve exibir um texto explícito indicando o tempo usado pelo seu algoritmo, especificando o segundo contêiner usado para ordenar a sequência de inteiros positivos.

O formato para a exibição do tempo usado para realizar sua ordenação é livre, mas a precisão escolhida deve permitir ver claramente a diferença entre os dois contêineres usados.

Aqui está um **exemplo** de uso padrão:

```
$> ./PmergeMe 3 5 9 7 4
Before:   3 5 9 7 4
After:    3 4 5 7 9
Time to process a range of      5 elements with std::[..] :     0.00031 us
Time to process a range of      5 elements with std::[..] :     0.00014 us
$> ./PmergeMe `shuf -i 1-100000 -n 3000 | tr "\n" " "`
Before:   141 79 526 321 [...]
After:    79 141 321 526 [...]
Time to process a range of   3000 elements with std::[..] :    62.14389 us
Time to process a range of   3000 elements with std::[..] :    69.27212 us
$> ./PmergeMe "-1" "2"
Error
$> # For OSX USER:
$> ./PmergeMe `jot -r 3000 1 100000 | tr '\n' ' '`
[...]
$>
```

A indicação do tempo é deliberadamente estranha neste exemplo. Claro que você tem que indicar o tempo usado para realizar todas as suas operações, tanto a parte de ordenação quanto a parte de gerenciamento de dados.

Aviso: O(s) contêiner(es) que você usou nos exercícios anteriores são proibidos aqui.

O gerenciamento de erros relacionados a duplicatas fica a seu critério.

# 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 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.

16D85ACC441674FBA2DF65190663F33F793984B142405F56715D5225FBAB6E3D6A4F
167020A16827E1B16612137E59ECD492E47AB764CB10B45D979615AC9FC74D521D9
20A778A5E