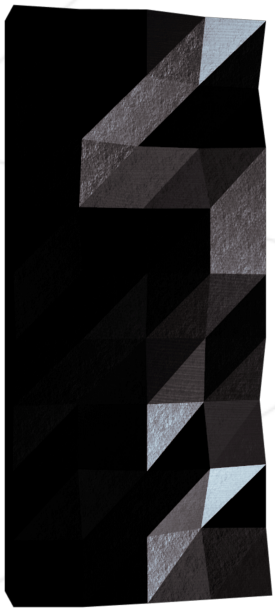

C++ - Módulo 02

Polimorfismo ad-hoc, sobrecarga de operadores e a Forma Canônica Ortodoxa da Classe

Preâmbulo: Este documento contém os exercícios do Módulo 02 dos módulos de C++.

Versão: 9.0

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 POO, mas decidimos escolher C++, pois é derivado do seu velho amigo C. Como esta é uma linguagem complexa, e para manter as coisas simples, o seu código deverá cumprir o padrão C++98.

Estamos cientes de que o C++ moderno é bastante diferente em muitos aspetos. Portanto, se você quiser 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

Novas regras

A partm ir de agora, todas as suas classes devem ser projetadas na **Forma Canônica Ortodoxa**, a menos que explicitamente declarado o contrário. Elas implementarão então as quatro funções de membro necessárias abaixo:

- Construtor padrão
- Construtor de cópia
- Operador de atribuição de cópia
- Destrutor

Divida o código da sua classe em dois arquivos. O arquivo de header (.hpp/.h) contém a definição da classe, enquanto o arquivo de origem (.cpp) contém a implementação.

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: Minha Primeira Classe na Forma Canônica Ortodoxa

	Exercice : 00
Minha Primeira Classe na Forma Canônica Ortodoxa	
Pasta de entrega : $ex00/$	
Arquivos para entregar : `Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp`	
Funções não permitidas : `Nenhuma`	

Você acha que conhece números inteiros e de ponto flutuante. Que bonitinho.

Por favor, leia este artigo de 3 páginas (1, 2, 3) para descobrir que você não conhece. Vá em frente, leia.

Até hoje, cada número que você usou no seu código era basicamente um inteiro ou um número de ponto flutuante, ou qualquer uma de suas variantes (`short`, `char`, `long`, `double` e assim por diante). Depois de ler o artigo acima, é seguro assumir que inteiros e números de ponto flutuante têm características opostas.

Mas hoje, as coisas vão mudar. Você vai descobrir um tipo de número novo e incrível: **números de ponto fixo**! Ausente para sempre dos tipos escalares da maioria das linguagens, números de ponto fixo oferecem um equilíbrio valioso entre desempenho, acurácia, alcance e precisão. Isso explica porque números de ponto fixo são particularmente aplicáveis a computação gráfica, processamento de som ou programação científica, apenas para citar alguns.

Como C++ carece de números de ponto fixo, você vai adicioná-los. Este artigo de Berkeley é um bom começo. Se você não tem ideia do que é a Universidade de Berkeley é, leia esta seção da sua página da Wikipedia.

Crie uma classe na Forma Canônica Ortodoxa que representa um número de ponto fixo:

- Membros privados:
  - Um **inteiro** para armazenar o valor do número de ponto fixo.
  - Um **inteiro constante estático** para armazenar o número de bits fracionários. Seu valor sempre será o literal inteiro `8`.
- Membros públicos:
  - Um construtor padrão que inicializa o valor do número de ponto fixo para `0`.
  - Um construtor de cópia.
  - Uma sobrecarga do operador de atribuição de cópia.
  - Um destrutor.
  - Uma função de membro `int getRawBits( void ) const;` que retorna o valor bruto do valor de ponto fixo.
  - Uma função de membro `void setRawBits( int const raw );` que define o valor bruto do número de ponto fixo.

Executando este código:

```
#include <iostream>

int		main( void ) {

  Fixed a;
  Fixed b( a );
  Fixed c;

  c = b;

  std::cout << a.getRawBits() << std::endl;
  std::cout << b.getRawBits() << std::endl;
  std::cout << c.getRawBits() << std::endl;

  return 0;
}
```

Deve produzir algo semelhante a:

```
$> ./a.out
Default constructor called
Copy constructor called
Copy assignment operator called   // <-- Esta linha pode estar faltando dependendo da sua implementação
getRawBits member function called
Default constructor called
Copy assignment operator called
getRawBits member function called
getRawBits member function called
0
getRawBits member function called
0
getRawBits member function called
0
```

```
Destructor called
Destructor called
Destructor called
$>
```

Capítulo VI

Exercício 01: Rumo a uma classe de número de ponto fixo mais útil

	Exercício : 01
Rumo a uma classe de número de ponto fixo mais útil	
Pasta de entrega : $ex01/$	
Arquivos para entregar : `Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp`	
Funções ou bibliotecas autorizadas : `roundf (da <cmath>)`	

O exercício anterior foi um bom começo, mas nossa classe é praticamente inútil. Ela só pode representar o valor `0.0`.

Adicione os seguintes construtores públicos e funções de membro públicas à sua classe:

- Um construtor que recebe um **inteiro constante** como um parâmetro.
  Ele o converte para o valor de ponto fixo correspondente. Os bits fracionários o valor deve ser inicializado para 8, como no exercício 00.
- Um construtor que recebe um **número de ponto flutuante constante** como um parâmetro.
  Ele o converte para o valor de ponto fixo correspondente. Os bits fracionários o valor deve ser inicializado para 8, como no exercício 00.
- Uma função de membro `float toFloat( void ) const;`
  que converte o valor de ponto fixo para um valor de ponto flutuante.
- Uma função de membro `int toInt( void ) const;`
  que converte o valor de ponto fixo para um valor inteiro.

E adicione a seguinte função aos arquivos da classe **Fixed**:

- Uma sobrecarga do operador de inserção («) que insere uma representação de ponto flutuante do número de ponto fixo no objeto de fluxo de saída passado como um parâmetro.

Executando este código:

```
#include <iostream>

int    main( void ) {

    Fixed       a;
    Fixed const b( 10 );
    Fixed const c( 42.42f );
    Fixed const d( b );

    a = Fixed( 1234.4321f );

    std::cout << "a is " << a << std::endl;
    std::cout << "b is " << b << std::endl;
    std::cout << "c is " << c << std::endl;
    std::cout << "d is " << d << std::endl;

    std::cout << "a is " << a.toInt() << " as integer" << std::endl;
    std::cout << "b is " << b.toInt() << " as integer" << std::endl;
    std::cout << "c is " << c.toInt() << " as integer" << std::endl;
    std::cout << "d is " << d.toInt() << " as integer" << std::endl;

    return 0;
}
```

Deve produzir algo semelhante a:

```
$> ./a.out
Default constructor called
Int constructor called
Float constructor called
Copy constructor called
Copy assignment operator called
Float constructor called
Copy assignment operator called
Destructor called
a is 1234.43
b is 10
c is 42.4219
d is 10
a is 1234 as integer
b is 10 as integer
c is 42 as integer
d is 10 as integer
Destructor called
Destructor called
Destructor called
Destructor called
$>
```

Capítulo VII

Exercício 02: Agora estamos falando

	Exercício : 02
Agora estamos falando	
Pasta de entrega : $ex02/$	
Arquivos para entregar : `Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp`	
Funções ou bibliotecas autorizadas : `roundf (da <cmath>)`	

Adicione funções de membro públicas à sua classe para sobrecarregar os seguintes operadores:

- Os 6 operadores de comparação: `>`, `<`, `>=`, `<=`, `==` e `!=`.
- Os 4 operadores aritméticos: `+`, `-`, `*` e `/`.
- Os 4 operadores de incremento/decremento (pré-incremento e pós-incremento, pré-decremento e pós-decremento), que aumentarão ou diminuirão o valor de ponto fixo pelo menor ϵ representável, de tal forma que $1 + \epsilon > 1$.

Adicione estas quatro funções de membro sobrecarregadas públicas à sua classe:

- Uma função de membro estática `min` que recebe duas referências a números de ponto fixo como parâmetros e retorna uma referência ao menor deles.
- Uma função de membro estática `min` que recebe duas referências a números de ponto fixo **constantes** como parâmetros e retorna uma referência ao menor deles.
- Uma função de membro estática `max` que recebe duas referências a números de ponto fixo como parâmetros e retorna uma referência ao maior deles.
- Uma função de membro estática `max` que recebe duas referências a números de ponto fixo **constantes** como parâmetros e retorna uma referência ao maior deles.

Cabe a você testar todos os recursos da sua classe. No entanto, executar o código abaixo:

```
#include <iostream>

int     main( void ) {

    Fixed          a;
    Fixed const    b( Fixed( 5.05f ) * Fixed( 2 ) );

    std::cout << a << std::endl;
    std::cout << ++a << std::endl;
    std::cout << a << std::endl;
    std::cout << a++ << std::endl;
    std::cout << a << std::endl;

    std::cout << b << std::endl;

    std::cout << Fixed::max( a, b ) << std::endl;

    return 0;
}
```

Deve produzir algo como (para maior legibilidade, o as mensagens do construtor/destrutor são removidas no exemplo abaixo):

```
$> ./a.out
0
0.00390625
0.00390625
0.00390625
0.0078125
10.1016
10.1016
$>
```

Se você alguma vez fizer uma divisão por 0, é aceitável que o programa falhe

Capítulo VIII

Exercício 03: BSP

	Exercício : 03
BSP	
Pasta de entrega : $ex03/$	
Arquivos para entregar : `Makefile, main.cpp, Fixed.{h, hpp}, Fixed.cpp, Point.{h, hpp}, Point.cpp, bsp.cpp`	
Funções ou bibliotecas autorizadas : `roundf (da <cmath>)`	

Agora que você tem uma classe **Fixed** funcional, seria bom usá-la.

Implemente uma função que indica se um ponto está dentro de um triângulo ou não. Muito útil, não é?

`BSP significa Binary Space Partitioning (Particionamento de Espaço Binário). De nada. :)`

`Você pode passar neste módulo sem concluir o exercício 03.`

Vamos começar criando a classe **Point** na Forma Canônica Ortodoxa que representa um ponto 2D:

- Membros privados:
  - Um atributo Fixed const `x`.
  - Um atributo Fixed const `y`.
  - Qualquer outra coisa útil.
- Membros públicos:
  - Um construtor padrão que inicializa `x` e `y` para `0`.
  - Um construtor que recebe dois números de ponto flutuante constantes como parâmetros. Ele inicializa `x` e `y` com esses parâmetros.
  - Um construtor de cópia.
  - Uma sobrecarga do operador de atribuição de cópia.
  - Um destrutor.
  - Qualquer outra coisa útil.

Para concluir, implemente a seguinte função no arquivo apropriado:

```
bool bsp( Point const a, Point const b, Point const c, Point const point);
```

- a, b, c: Os vértices do nosso amado triângulo.
- point: O ponto a ser verificado.
- Retorna: True se o ponto estiver dentro do triângulo. False caso contrário.
  Assim, se o ponto for um vértice ou estiver em uma aresta, ele retornará False.

Implemente e entregue seus próprios testes para garantir que sua classe se comporte como esperado.

Capítulo IX

Entrega e Avaliação por Pares

Entregue sua tarefa em seu repositório `Git` como de costume. Apenas o 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 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.

```
???????????  XXXXXXXXXX = $3$$d6f957a965f8361750a3ba6c97554e9f
```