

42 (../)

ÿ

ferrugem

## ESCALA PARA PROJETO CPP MÓDULO 02 (HTTPS:// PROJECTS.INTRA.42.FR/PROJECTS/ CPP-MODULE-02 )

Você deve avaliar 1 aluno nesta equipe

### Introdução

Por favor, respeite as seguintes regras:

- Permaneça educado, cortês, respeitoso e construtivo durante o processo de avaliação. O bem-estar da comunidade depende disso.
- Identificar junto ao avaliado ou ao grupo avaliado possíveis disfunções em seu trabalho. Aproveite o tempo para discuti-lo e debater os problemas identificados.
- Deve ter em conta que podem existir ligeiras diferenças de interpretação entre as instruções do projeto, o seu âmbito e as suas funcionalidades. Mantenha a mente aberta e avalie o mais honestamente possível. A pedagogia só é válida se a avaliação pelos pares for feita com seriedade.

### Diretrizes

- Anote apenas o que está contido no repositório Git clonado do aluno ou grupo.
- Verifique se o repositório Git pertence ao aluno ou grupo, se o projeto é o esperado e se o "git clone" é utilizado em uma pasta vazia.
- Verifique cuidadosamente se nenhum pseudônimo foi usado para enganá-lo e certifique-se de avaliar a tradução oficial corretamente.
- Para evitar surpresas, verifique com o aluno ou grupo sobre possíveis roteiros utilizados para facilitar a avaliação (por exemplo, roteiros de teste ou automação).
- Caso ainda não tenha feito o projeto que vai avaliar, deverá ler todo o tópico antes de iniciar a avaliação.
- Use os flags disponíveis para sinalizar uma renderização vazia, um programa

não funciona, erro padrão, trapaça... Nessas situações, a avaliação é finalizada e a pontuação é 0, ou -42 em caso de trapaça. Porém, com exceção dos casos de trapaça, incentivamos você a continuar a discussão sobre o trabalho submetido, mesmo que este esteja incompleto. Isso serve para identificar erros que não devem ser reproduzidos no futuro.

- Se o tópico exigir um arquivo de configuração, você nunca deverá modificá-lo. Se você deseja editar um arquivo, explique o motivo à pessoa que está sendo avaliada e certifique-se de ter seu consentimento.

- Você também deve verificar se há vazamentos de memória. Toda a memória alocado no heap deve ser liberado adequadamente antes do final da execução do programa.

Você tem o direito de usar qualquer ferramenta disponível na máquina, como vazamentos, valgrind ou e\_fence. Em caso de vazamento de memória, verifique o sinalizador apropriado.

---

## Anexos

assunto.pdf (<https://github.com/rphlr/42-Subjects/>)

## Testes preliminares

Se houver suspeita de trapaça, a classificação e avaliação terminam imediatamente. Para denunciar isso, selecione o sinalizador "Cheat". Tenha o cuidado de usá-lo com calma, cuidado e discernimento.

---

### Pré-requisitos

O código deve ser compilado com C++ e as flags -Wall -Wextra -Werror Lembrando que este projeto deve seguir o padrão C++98. Portanto, funções e contêineres C++ 11 (ou outro padrão) NÃO são esperados.

Não avalie o exercício se você encontrar:

- Uma função implementada em um arquivo de cabeçalho (exceto funções de modelo).
- Um Makefile compilando sem os flags solicitados e/ou com algo diferente de c++.

Selecione o sinalizador "Função proibida" se você conhece :

- O uso de uma função "C" (\*alloc, \*printf, free).
- A utilização de função proibida no projeto.
- O uso de "using namespace <ns\_name>" ou da palavra-chave "friend".
- O uso de uma biblioteca externa ou recursos específicos da versão posterior ao C++98.

Sim

Não

## Exercício 00: Meu primeiro canhão

Este exercício introduz a noção de classe canônica com um exercício aritmético simples: números de ponto fixo.

### Makefile

Existe um Makefile que compila usando os sinalizadores apropriados.

Sim

Não

### Acessadores

A classe Fixa (ou outra) deve ter acessadores para o valor bruto:

- `int getRawBits( void ) const;` •  
`void setRawBits(int const raw);` Esses membros estão presentes e funcionais?

Sim

Não

### Classe canônica

Uma classe canônica deve ter pelo menos:

- Um construtor padrão • Um destruidor
  - Um construtor por cópia • Um operador de atribuição
- Esses elementos estão presentes e funcionais?

Sim

Não

## Exercício 01: Primeiros passos para uma aula útil

O exercício anterior foi um bom primeiro passo. Porém, a classe era de pouca utilidade, pois só permitia representar o valor 0,0.

### Makefile

Existe um Makefile que compila usando os sinalizadores apropriados.

Sim

Não

### Construtor via flutuante

Podemos construir uma instância a partir de um número de ponto flutuante?

Sim

Não

### operador <<

Existe um operador << e ele é funcional?

Sim

Não

### Ponto fixo para inteiro

A classe deve incluir uma função de membro "int toInt(void) const;"  
que converte um número de ponto fixo em um int.

Está presente e funcional?

Sim

Não

### Ponto fixo para flutuar

A classe deve incluir uma função de membro "float toFloat(void) const;"  
que converte um número de ponto fixo em um número flutuante.

Está presente e funcional?

Sim

Não

### Construção com um int

Podemos instanciar a classe com o construtor recebendo um int?

Sim

Não

## Exercício 02: Agora podemos conversar

Este exercício adiciona os operadores de comparação e aritméticos à classe.

### Makefile

Existe um Makefile que compila usando os sinalizadores apropriados.

Sim

Não

### Operadores de comparação

Os 6 operadores de comparação (>, <, >=, <=, == e !=) estão presentes e funcionais?

Sim

Não

---

### Operadores aritméticos

Os 4 operadores aritméticos (+, -, (se \* e/) estão presentes e funcionais?  
você dividir por 0, é aceitável que o programa trave).

Sim

Não

---

### Outras operadoras

Os quatro operadores de incremento e decremento  
(pré-incremento e pós-incremento, pré-decremento e  
pós-decremento) eles estão presentes e funcionais?

Sim

Não

---

### Sobrecarregando funções de membros estáticos públicos

Finalmente, verifique se o membro estático funciona min() e max()  
estão implementados e funcionais.

Sim

Não

---

## Exercício 03: BSP

Este exercício deve fazer você perceber como é fácil implementar  
algoritmos complexos, uma vez que o básico funciona conforme o esperado.

---

### Makefile

Existe um Makefile que compila usando os sinalizadores apropriados.

Sim

Não

---

### Ponto de aula

Existe uma classe Point que possui dois atributos (x e y) do tipo  
Const fixo.

Ele também possui um construtor que usa dois floats e  
inicializa x e y com esses valores.

Sim

Não

---

### função bsp

Existe uma função bsp() cujo protótipo é

"bool bsp (Ponto const a, Ponto const b, Ponto const c, Ponto const ponto)".

A função retorna True se o ponto estiver dentro do triângulo descrito por vértices a, b e c. Caso contrário, retorna False.

Sim

Não

Principal e testes

Existe um método principal para testar se a função bsp() funciona conforme descrito acima. Execute vários testes para garantir que o valor de o retorno está correto.

Sim

Não

## Avaliações

Não se esqueça de verificar a bandeira correspondente à defesa

OK

Projeto excelente

Trabalho vazio

Trabalho incompleto W Compilação inválida

Trair

Colidir


Em relação à situação



Vazamentos

I Função proibida

Não é possível apoiar/explicar o código

Dê uma estrela a este repositório. 

(<https://github.com/rphlr/42-Evals>)