

ESCALA PARA PROJETO CPP MÓDULO 01 (HTTPS:// PROJECTS.INTRA.42.FR/ PROJECTS/ CPP-MODULE-01)

Você deve avaliar 1 aluno nesta equipe

Introdução

Por favor, cumpra as seguintes regras:

- Permaneça educado, cortês, respeitoso e construtivo durante todo o processo de avaliação. O bem-estar da comunidade depende disso.
- Identificar junto ao aluno ou grupo cujo trabalho é avaliado as possíveis disfunções no seu projeto. Aproveite o tempo para discutir e debater os problemas que possam ter sido identificados.
- Você deve considerar que pode haver algumas diferenças na forma como seus pares podem ter entendido as instruções do projeto e o escopo de suas funcionalidades. Sempre mantenha a mente aberta e avalie-os da forma mais honesta possível. A pedagogia só é útil e somente se a avaliação pelos pares for feita com seriedade.

Diretrizes

- Avaliar somente o trabalho que foi entregue no repositório Git do aluno ou grupo avaliado.
- Verifique novamente se o repositório Git pertence ao(s) aluno(s). Certifique-se de que o projeto é o esperado. Além disso, verifique se 'git clone' é usado em uma pasta vazia.
- Verifique cuidadosamente se nenhum aliase malicioso foi usado para enganar-lo e fazer com que você avalie algo que não seja o conteúdo do repositório oficial.
- Para evitar surpresas e se for o caso, revisem juntos quaisquer scripts utilizados para facilitar a classificação (scripts para testes ou automação).
- Caso não tenha concluído o trabalho que vai avaliar, deverá ler a matéria na íntegra antes de iniciar o processo de avaliação.
- Use os sinalizadores disponíveis para relatar um repositório vazio, um programa que não funciona, um erro de norma, trapaça e assim por diante.

Nestes casos, o processo de avaliação termina e a nota final é 0, ou -42 em caso de trapaça. Porém, exceto em caso de trapaça, os alunos são fortemente incentivados a revisarem juntos o trabalho entregue, a fim de identificar eventuais erros que não devam ser repetidos no futuro.

- Você nunca deverá editar nenhum arquivo, exceto o arquivo de configuração, se ele existir. Se você quiser editar um arquivo, reserve um tempo para explicar os motivos ao aluno avaliado e certifique-se de que ambos concordam com isso.

- Você também deve verificar a ausência de vazamentos de memória. Qualquer memória alocada no heap deve ser liberada adequadamente antes do final da execução.

Você tem permissão para usar qualquer uma das diferentes ferramentas disponíveis no computador, como leaks, valgrind ou e_fence. Em caso de vazamento de memória, marque o sinalizador apropriado.

Anexos

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

Testes preliminares

Se houver suspeita de trapaça, a avaliação termina aqui. Use o sinalizador "Cheat" para denunciá-lo. Tome esta decisão com calma, sabedoria e, por favor, use este botão com cautela.

Pré-requisitos

O código deve ser compilado em c++ e as flags -Wall -Wextra -Werror Não se esqueça que este projeto deve seguir o padrão C++98. Portanto, funções ou contêineres C++ 11 (e posteriores) NÃO são esperados.

Qualquer um destes significa que você não deve avaliar o exercício em questão:

- Uma função é implementada em um arquivo de cabeçalho (exceto funções de modelo).
- Um Makefile compila sem os sinalizadores necessários e/ou outro compilador que não seja c++.

Qualquer um destes significa que você deve sinalizar o projeto com "Proibido Função":

- Utilização de uma função "C" (*alloc, *printf, free).
- Utilização de função não permitida nas orientações do exercício.
- Uso de "using namespace <ns_name>" ou da palavra-chave "friend".
- Utilização de biblioteca externa ou recursos de versões diferentes de C++98.

Sim

Não

Exercício 00: BraiiiiiinnnnzzzzZ

O objetivo deste exercício é entender como alocar memória em C++.

Makefile e testes

Existe um Makefile que compila usando os sinalizadores apropriados.

Existe pelo menos um main para testar o exercício.

Sim

Não

Classe Zumbi

Existe uma classe de zumbis.

Possui um atributo de nome privado.

Tem pelo menos um construtor.

Possui uma função membro anuncia(void) que imprime: "<nome>: BraiiiiiiinnzzzzZ..."

O destruidor imprime uma mensagem de depuração que inclui o nome do zumbi.

Sim

Não

novoZumbi

Existe uma função newZombie() prototipada como: [Zombie* newZombie(std::string name);]

Deve alocar um Zumbi na pilha e devolvê-lo.

Idealmente, ele deveria chamar o construtor que pega uma string e inicializa o nome.

O exercício deve ser marcado como correto se o Zumbi puder se anunciar com o nome passado para a função.

Existem testes para comprovar que tudo funciona.

O zumbi é excluído corretamente antes do final do programa.

Sim

Não

aleatórioChump

Existe uma função randomChump() prototipada como: [void randomChump(std::string name);]

Deve criar um Zumbi na pilha e fazê-lo se anunciar.

Idealmente, o zumbi deve ser alocado na pilha (excluído implicitamente no final da função).

Ele também pode ser alocado no heap e depois excluído explicitamente.

O aluno deve justificar suas escolhas.

Existem testes para comprovar que tudo funciona.

Sim

Não

Exercício 01: Moar brainz!

O objetivo deste exercício é alocar vários objetos ao mesmo tempo usando new[], inicializá-los e excluí-los adequadamente.

Makefile e testes

Existe um Makefile que compila usando os sinalizadores apropriados.

Existe pelo menos um main para testar o exercício.

Sim

Não

horda de zumbis

A classe Zombie possui um construtor padrão.

Existe uma função zombieHorde() prototipada como: [Zombie* zombieHorde(int N, std::string name);]

Ele aloca N zumbis no heap explicitamente usando new[].

Após a alocação, ocorre uma inicialização dos objetos para definição de seus nomes.

Ele retorna um ponteiro para o primeiro zumbi.

Existem testes suficientes para comprovar os pontos anteriores.

Tipo: chamar anúncio() em todos os zumbis.

Por último, todos os zumbis devem ser excluídos ao mesmo tempo no modo principal.

Sim

Não

Exercício 02: OI, ISSO É CÉREBRO

Desmistifique as referências! Desmistifique as referências! Desmistifique as referências! Desmistifique as referências!

Desmistifique as referências! Desmistifique as referências! Desmistifique as referências! Desmistifique as referências!

Desmistifique as referências! Desmistifique as referências! Desmistifique as referências! Desmistifique as referências!

Makefile e testes

Existe um Makefile que compila usando os sinalizadores apropriados.

Existe pelo menos um main para testar o exercício.

Sim

Não

OI, ISSO É CÉREBRO

Há uma string contendo "HI THIS IS BRAIN". stringPTR é um

ponteiro para a string. stringREF é uma

referência à string.

O endereço da string é exibido usando a variável string, stringPTR e stringREF.

O conteúdo da variável é exibido usando stringPTR e stringREF.

Sim

Não

Exercício 03: Violência desnecessária

O objetivo deste exercício é entender que ponteiros e referências apresentam algumas pequenas diferenças que os tornam menos ou mais apropriados dependendo do uso e do ciclo de vida do objeto utilizado.

Makefile e testes

Existe um Makefile que compila usando os sinalizadores apropriados.

Existe pelo menos um main para testar o exercício.

Sim

Não

Arma

Existe uma classe Weapon que possui uma string de tipo, um getType() e um setType().

A função getType() retorna uma referência const para o tipo string.

Sim

Não

HumanA e HumanB

HumanA pode ter uma referência ou um ponteiro para a Arma.

O ideal é que seja implementada como referência, já que a Arma existe desde a criação até a destruição, e nunca muda.

HumanB deve ter um ponteiro para uma Arma, pois o campo não está definido no momento da criação e a arma pode ser NULL.

Sim

Não

Exercício 04: Sed é para perdedores

Graças a este exercício, o aluno deverá ter se familiarizado com ifstream e ofstream.

Makefile e testes

Existe um Makefile que compila usando os sinalizadores apropriados.

Existe pelo menos um main para testar o exercício.

Sim

Não

Exercício 04

Existe uma função replace (ou outro nome) que funciona conforme especificado no assunto.

O gerenciamento de erros é eficiente: tente passar um arquivo que não existe, altere as permissões, passe vazio, etc.

Se você encontrar um erro que não é tratado e que não é completamente esotérico, não há pontos para este exercício.

O programa deve ler o arquivo usando um `ifstream` ou equivalente e escrever usando um `ofstream` ou equivalente.

A implementação da função deve ser feita usando funções de `std::string`, não lendo a string caractere por caractere.

Isso não é mais C!

Sim

Não

Exercício 05: Harl 2.0

O objetivo deste exercício é usar ponteiros para funções de membros de classe. Além disso, esta é a oportunidade de descobrir os diferentes níveis de log.

Makefile e testes

Existe um Makefile que compila usando os sinalizadores apropriados.

Existe pelo menos um main para testar o exercício.

Sim

Não

Nosso querido Harl

Existe uma classe Harl com pelo menos as 5 funções exigidas na disciplina.

A função `reclamar()` executa as outras funções usando um ponteiro para elas.

Idealmente, o aluno deveria ter implementado uma forma de combinar as diferentes strings correspondentes ao nível de log com os ponteiros da função membro correspondente.

Se a implementação for diferente, mas o exercício funcionar, você deverá marcá-lo como válido. A única coisa que não é permitida é usar um `if/elseif/else` feio.

O aluno poderia ter optado por alterar a mensagem que Harl exibe ou exibir os exemplos dados na disciplina, ambos são válidos.

Sim

Não

Exercício 06: Filtro Harl

Agora que você é um programador experiente, você deve usar novos tipos de instruções, instruções, loops, etc.

O objetivo deste último exercício é fazer com que você descubra a instrução `switch`.

Makefile e testes

Existe um Makefile que compila usando os sinalizadores apropriados.

Existe pelo menos um main para testar o exercício.

Sim

Não

Desligando Harl

O programa harlFilter toma como argumento qualquer um dos níveis de log ("DEBUG", "INFO", "AVISO" ou "ERRO"). Deve então exibir apenas as mensagens que estão no mesmo nível ou acima (DEBUG < INFO < WARNING < ERROR). Esse deve ser implementado usando uma instrução switch com um caso padrão. Mais uma vez, não há mais if/elseif/else, por favor.

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

Situação preocupante

 Vazamentos

I Função proibida

Não é possível apoiar/explicar o código

Dê uma estrela a este repositório. 

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