(../) ÿ **Rouille**

ESCALA PARA PROJETO CPP MÓDULO 08

(HTTPS://PROJECTS.INTRA.42.FR/PROJECTS/CPP- MODULE-08)

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 entenderam 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 enganá-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":

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

Sim Não

Ex00: Fácil localização

Como sempre, deve haver uma função principal que contenha testes suficientes para provar que o programa funciona conforme o esperado. Se não houver, não avalie este exercício. Se alguma classe sem interface não estiver na forma de classe canônica ortodoxa, não avalie este exercício.

Função de modelo

Existe uma função modelada easyFind(T, int) que faz o que o assunto exige.

TEM que usar algoritmos STL.
Caso contrário (como pesquisa manual usando iteradores, por exemplo), considere-o errado.

Sim Não

Ex01: Período

Como sempre, deve haver uma função principal que contenha testes suficientes para provar que o programa funciona conforme o esperado. Se não houver, não avalie este exercício. Se alguma classe sem interface não estiver na forma de classe canônica ortodoxa, não avalie este exercício.

Funções de classe e membro

Existe uma classe que atende às restrições da disciplina.
Suas funções-membro usam algoritmos STL para encontrar seus resultados tanto quanto possível. Encontrar o intervalo mais curto não pode ser feito apenas subtraindo os dois números mais baixos (dê uma olhada no exemplo do assunto).

Sim Não

Função addNumber aprimorada

Existe uma maneira de adicionar números que é mais prática do que chamar a função addNumber() repetidamente.

Sim Não

Ex02: Abominação mutante

Como sempre, deve haver uma função principal que contenha testes suficientes para provar que o programa funciona conforme o esperado. Se não houver, não avalie este exercício. Se alguma classe sem interface não estiver na forma de classe canônica ortodoxa, não avalie este exercício.

Classe MutantStack

Existe uma classe MutantStack que herda de std::stack e oferece todas as suas funções-membro.
Tem um iterador. Além disso, é possível fazer pelo menos as operações mostradas nos exemplos da disciplina usando iteradores.

Sim Não

Melhores testes

Existe pelo menos uma função main() que tem mais testes do que aqueles do assunto.

Sim Não

Avaliações

Não se esqueça de verificar a bandeira correspondente à defesa

OK Projeto excelente

Trabalho vazio Trabalho incompleto Compilação inválida Trair Colidir

Situação preocupante Vazamentos l Função proibida

Dê uma estrela a este repositório.

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