Aluno: Gabriel Leite Queiroz

Matéria: Qualidade de Software e Governança

Curso: Análise e Desenvolvimento de Sistemas

Relatório de Refatoração e Análise de Código

Descrição do Projeto

O código gerencia processos usando duas estruturas de dados: fila (FIFO) e pilha (LIFO). O programa permite ao usuário:

- Adicionar processos à fila.
- Remover processos da fila e empilhá-los.
- Empilhar processos diretamente.
- Desempilhar processos e mostrar seus IDs.
- Exibir o histórico de processos desempilhados.

A interação é feita por meio de um menu onde o usuário escolhe as operações, e o sistema gerencia os processos com base nas escolhas. No entanto, o código apresenta problemas de lógica, estilo e verificações inadequadas que precisam ser corrigidos para melhorar sua eficiência e segurança.

Relatório de Qualidade de Código

Complexidade Ciclomática: O código apresenta uma complexidade ciclômática moderada, com várias funções que envolvem controle de fluxo (condições e laços), como no switch no main e nas funções enqueue, dequeue, push, e pop. Cada função possui um único ponto de decisão, mas o fluxo do código pode ser mais bem simplificado em alguns casos.

Cobertura de Código: O código não possui testes unitários ou cobertura explícita. Cada funcionalidade (fila e pilha) é chamada diretamente a partir do main, mas não há verificação de todos os cenários possíveis, como erro de alocação de memória ou falhas na manipulação dos ponteiros.

Dívida Técnica: Há uma dívida técnica significativa devido à má alocação de memória e à falta de gerenciamento adequado dos ponteiros front e rear na fila. Além disso, não há liberação de memória no pop da pilha, o que pode levar a vazamentos de memória.

Código Duplicado: Não há duplicação explícita de código, mas funções similares como enqueue e dequeue e push e pop podem ser refatoradas para compartilhar mais lógica.

Más Práticas:

Verificação de erros: Falta verificação adequada de erro, como quando a alocação de memória falha em malloc.

Compatibilidade: A função clear_screen usa system("clear"), o que não é portátil, pois não funciona no Windows.

Formatação: A função print_history apresenta a lista de processos sem a formatação adequada, como um espaço após o ID do processo.

Liberação de memória: No pop, a memória não é liberada, o que pode causar vazamentos de memória.

Vulnerabilidades de Segurança:

Uso do system("clear"): Isso pode ser vulnerável a injeção de comandos, especialmente se os dados forem manipulados incorretamente.

Falha ao verificar erro de malloc: A ausência de checagem após malloc pode permitir que o programa continue executando mesmo após uma falha de alocação de memória.

Plano de Refatoração:

Melhorar Verificação de Erros: Adicionar verificações de falhas em malloc.

Corrigir Gerenciamento de Ponteiros: Ajustar as funções de fila e pilha para garantir a atualização correta dos ponteiros e liberar memória em pop.

Ajustar Formatação de Impressão: Melhorar a apresentação do histórico de processos.

Remover Dependência de system("clear"): Evitar comandos específicos do sistema operacional.

Refatorar Fluxo de Controle: Consolidar verificações de fila e pilha vazias em funções auxiliares.

Adicionar Comentários: Incluir explicações no código para melhorar a legibilidade.