Universidade Federal de Mato Grosso

Bacharelado em Ciência da Computação

Semestre: 2024/1

Programação 2

Lista 3

Descrição do Problema

Desenvolva um sistema de gerenciamento de usuários utilizando ponteiros inteligentes em C++. Cada usuário possui um nome e pode ter dependências para outros usuários. Utilize `std::shared_ptr` para gerenciar a alocação de usuários compartilhados e `std::weak_ptr` para representar dependências, evitando ciclos de referência. Além disso, cada usuário pode possuir um recurso exclusivo gerenciado por `std::unique_ptr`.

Tarefas

1. Definir a estrutura `struct Usuario` com membros para o nome, dependências (usando `std::vector<std::weak_ptr<Usuario>>`) e um recurso exclusivo (usando `std::unique_ptr<int>`).
2. Implementar funções para:
   - Adicionar um novo usuário ao sistema.
   - Adicionar uma dependência a um usuário existente.
   - Listar todas as dependências de um usuário específico.
   - Atribuir um valor ao recurso exclusivo de um usuário.
   - Encontrar um usuário pelo nome para ser utilizado nas operações acima.
3. Escrever um programa principal (`main`) que permita ao usuário interagir com o sistema de usuários, adicionando usuários, criando dependências entre eles, listando as dependências de um usuário e atribuindo valores aos recursos exclusivos.

Explicação

Este exercício visa praticar o uso de ponteiros inteligentes (`std::shared_ptr`, `std::weak_ptr` e `std::unique_ptr`) em C++ para gerenciar recursos de memória de forma eficiente e segura. O uso de `std::shared_ptr` permite o compartilhamento seguro de usuários entre várias dependências, enquanto `std::weak_ptr` é utilizado para representar dependências sem criar ciclos de referência que poderiam levar a vazamentos de memória. Por fim, `std::unique_ptr` é utilizado para garantir a exclusividade de recursos associados a cada usuário, evitando duplicação ou vazamentos de memória.

Estrutura de Dados e Protótipos

Struct Usuario

```
struct Usuario {
    std::string nome;
    std::vector<std::weak_ptr<Usuario>> dependencias;
    std::unique_ptr<int> recursoExclusivo;
};
```

Protótipos das Funções

```
void adicionarUsuario(const std::string& nome);
void adicionarDependencia(const std::string& nomeUsuario, const std::
   string& nomeDependencia);
void listarDependencias(const std::string& nomeUsuario);
void atribuirRecursoExclusivo(const string& nomeUsuario, int valor);
std::shared_ptr<Usuario> encontrarUsuario(const std::string& nome);
```

Vetor global de usuários (para uso em adicionarUsuario)

```
std::vector<std::shared_ptr<Usuario>> usuarios;
```

Programa Principal

```
int main() {
    adicionarUsuario("Alice");
    adicionarUsuario("Bob");
    adicionarUsuario("Carol");

    adicionarDependencia("Alice", "Bob");
    adicionarDependencia("Alice", "Carol");

    atribuirRecursoExclusivo("Alice", 42);

    listarDependencias("Alice");
    listarDependencias("Bob");

    return 0;
}
```

Saída Esperada

```
Usuario 'Alice' adicionado.
Usuario 'Bob' adicionado.
Usuario 'Carol' adicionado.
Usuario 'Bob' adicionado como dependencia de 'Alice'.
Usuario 'Carol' adicionado como dependencia de 'Alice'.
Recurso exclusivo de 'Alice' atribuído com valor: 42.
Dependências de 'Alice':
- Bob
- Carol
Dependências de 'Bob':
```