Linguagem de Programação (LPG0002)

Sistema de Agendamento de Salas

Alunos: Guilherme Ytalo e Nícolas Morais

3. Objetivos

O objetivo principal deste trabalho foi desenvolver um sistema em linguagem C para gerenciar o agendamento de salas na universidade. O sistema permite listar salas, listar reservas, agendar salas e cancelar reservas. A aplicação foi projetada para otimizar o uso das salas e evitar conflitos de agendamento.

4. Desenvolvimento da atividade

Objetivos e Estratégias

Identificação dos Objetivos

1. Criar um sistema de gerenciamento de salas.
2. Garantir que o sistema seja intuitivo e fácil de usar.
3. Armazenar os dados de forma persistente.

Etapas de Desenvolvimento

1. **Análise de Requisitos**: Identificar as funcionalidades necessárias para o sistema.
2. **Projeto do Sistema**: Definir a estrutura dos dados (salas e reservas) e planejar as funções necessárias.
3. **Implementação**: Escrever o código em C, utilizando um arquivo principal (`main.c`) e módulos separados para cada funcionalidade.
4. **Teste e Validação**: Testar cada função do sistema para garantir que todas as operações (agendamento, cancelamento, listagem) funcionam corretamente.

Implementação

Estrutura dos Dados

Os dados foram estruturados em dois tipos principais:

- `Sala`: Representa uma sala, com campos como ID, nome, descrição, lotação máxima e disponibilidade.
- `Reserva`: Representa uma reserva, associando uma sala a uma data específica e um número de participantes.

Funções Implementadas

1. **Inicialização das Salas** (`inicializarSalas`): Carrega uma lista pré-definida de salas.
2. **Criação de Sala** (`criarSala`): Permite ao usuário adicionar novas salas ao sistema.
3. **Reserva de Sala** (`reservarSala`): Garante que uma sala esteja disponível antes de permitir a reserva.
4. **Listagem de Salas** (`listarTodasSalas`): Exibe todas as salas e suas respectivas informações.
5. **Listagem de Reservas** (`listarTodasReservas`, `listarReservasDeUmaSala`): Exibe todas as reservas feitas ou as reservas de uma sala específica.
6. **Cancelamento de Reserva** (`cancelarReserva`): Permite ao usuário cancelar uma reserva existente.
7. **Encerramento do Programa** (`encerrarPrograma`): Libera a memória alocada e encerra a execução do programa.

Estratégias de Solução

1. **Modularização**: O código foi dividido em múltiplos arquivos (`.c` e `.h`) para organizar as diferentes funcionalidades do sistema.
2. **Persistência de Dados**: Utilização de arquivos para armazenar as informações de salas e reservas, garantindo que os dados permaneçam após o encerramento do programa.
3. **Verificação de Disponibilidade**: Implementação de uma função específica para verificar a disponibilidade das salas antes de confirmar uma reserva.

Dificuldades Encontradas

1. **Gerenciamento de Memória**: A alocação dinâmica de memória para as listas de salas e reservas exigiu atenção especial para evitar vazamentos de memória e garantir a integridade dos dados.
2. **Verificação de Disponibilidade**: Implementar uma função eficiente para verificar a disponibilidade das salas em datas específicas foi um desafio, especialmente ao lidar com diferentes formatos de datas.

5. Considerações finais

O desenvolvimento deste sistema proporcionou uma oportunidade valiosa para aplicar na prática o que aprendemos em sala de aula, como manipulação de strings, alocação dinâmica de memória e manipulação de arquivos. Através deste trabalho, foi possível criar uma solução prática para um problema real, ao mesmo tempo em que se adquiriu uma maior compreensão sobre a importância de um design de software bem estruturado e modularizado. As dificuldades encontradas ao longo do projeto foram superadas com pesquisa e prática, resultando em um sistema funcional e eficiente para o agendamento de salas.

6. Referências

- H. M. Deitel, P. J. Deitel Bookman. **C Como Programar**. 6. ed Pearson, 2011.
- Conteúdos disponibilizados no site do professor.
- ricardofm.net