Desafio de Estruturas em C: Gerenciamento de Quartos de Hotel

Minicurso de C - PET EEL

September 1, 2025

1 Objetivo do Desafio

O objetivo deste desafio é desenvolver um programa em C que gerencie a ocupação de quartos de um hotel. O foco principal é praticar o uso de:

- **Structs** para organizar os dados de cada quarto;
- **Vetores (arrays)** para armazenar todos os quartos;
- **Union** para serviços opcionais, economizando memória;
- Lógica de programação para inserir, listar e alterar informações.

Não será necessário usar `malloc` ou alocação dinâmica, todo o vetor será fixo.

2 Struct Sugerida

A struct que vocês devem usar para representar cada quarto é:

```
typedef struct {
    int numero;              // N mero do quarto
    char nome[50];           // Nome do h spede
    char tipo[20];           // Tipo do quarto (Simples, Luxo, Suite)
    union {
        int cafeDaManha;     // 0 = n o, 1 = sim
        int estacionamento; // 0 = n o, 1 = sim
        int spa;             // 0 = n o, 1 = sim
    } servicos;
    int checkin[2];          // [dia, hora]
    int checkout[2];         // [dia, hora]
} quarto_t;
```

O vetor de quartos será definido assim:

```
#define MAX_QUARTOS 5
quarto_t hotel[MAX_QUARTOS];
```

—

3 Tarefas do Programa

O programa deve permitir:

1. **Inserir hóspede:** preencher os dados de um quarto (número, nome, tipo, serviço, check-in e check-out).
2. **Listar quartos:** mostrar todas as informações cadastradas.
3. **Alterar quarto:** atualizar os dados de um quarto específico.

4 Exemplo de Saída Esperada

```
1. Inserir hóspede
2. Listar quartos
3. Alterar quarto
4. Sair
Escolha uma opção: 1

Digite o número do quarto: 101
Nome do hóspede: Thiago
Tipo do quarto: Luxo
Serviço (1-Café, 2-Estac., 3-Spa): 2
Check-in (dia hora): 1 14
Check-out (dia hora): 5 12

Quarto cadastrado com sucesso!

--- Listando quartos ---
Quarto 101: Thiago, Luxo
Serviço ativo: Estacionamento
Check-in: dia 1 hora 14
Check-out: dia 5 hora 12
```

—

5 Dicas de Lógica

Para ajudar na implementação, seguem algumas dicas de como pensar a lógica:

- **Passo 1: Inicializar o vetor** Antes de inserir qualquer hóspede, inicialize os campos do vetor de quartos para valores padrão (ex.: números negativos, strings vazias, serviços 0).
- **Passo 2: Inserir Hóspede** Solicite ao usuário as informações do hóspede e preencha o struct correspondente no vetor. Lembre-se de acessar o quarto pelo índice correto.
- **Passo 3: Listar Quartos** Percorra todo o vetor e mostre apenas os quartos já cadastrados (ex.: aqueles com número diferente de zero ou nome não vazio).
- **Passo 4: Alterar Quarto** Pergunte qual quarto o usuário quer alterar, procure pelo número do quarto e permita atualizar os campos desejados.

- **Passo 5: Serviços com Union** Use apenas um campo da union por vez. Se o usuário escolher "Estacionamento", não é necessário preencher "Café" ou "Spa" para aquele quarto.
- **Passo 6: Check-in e Check-out** Armazene dia e hora em uma matriz de 2 elementos dentro do struct (`int checkin[2]`).
- **Dicas gerais de C:**
  - Use `strcpy()` para strings.
  - Use vetores e índices cuidadosamente.
  - Teste cada função individualmente antes de integrar tudo.

—

6 Sugestão de Organização do Código

1. **main()** com menu de opções.
2. Função `inserirHospede(quarto_t hotel[], int indice)`.
3. Função `listarQuartos(quarto_t hotel[], int total)`.
4. Função `alterarQuarto(quarto_t hotel[], int indice)`.
5. (Opcional) Função `inicializarHotel(quarto_t hotel[], int total)` para limpar o vetor.

—

7 Conclusão

Este desafio permite praticar:

- Uso de **structs** e **unions**;
- Vetores de structs;
- Lógica de inserção, listagem e atualização de dados;
- Manipulação de strings e números;
- Pensamento organizado passo a passo.