Relatorio trabalho final LPG

Arquivos de Cabeçalho e Definições:

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#define RED   "\x1B[31m"

#define RESET "\x1B[0m"

#define MAX_AMBIENTES 10
```

Arquivos de Cabeçalho: Incluem arquivos de cabeçalho necessários para entrada/saída padrão, alocação dinâmica de memória, manipulação de strings e configurações de localidade.

Definições: Define códigos de cores para formatação de saída (RED para texto vermelho e RESET para resetar a formatação). Também define o número máximo de entradas de ambientes (MAX_AMBIENTES) como 10.

Estruturas de Dados: Define estruturas para representar data (struct Data), intervalo de tempo (struct Horario), eventos individuais (struct evento) e ambientes/recursos associados a eventos (struct ambiente)

```
struct Data {

   int dia, mes;

};

struct Horario {

   int hora1, min1, hora2, min2;

};

struct evento {
```

```
    struct Data data;
    struct Horario horario;
    char nomeEvento[50];
    char salaEvento[20];
    char descricao[20];
};
struct ambiente {
    struct evento x;
    char local[20];
};
```

Funções de Validação: Essas funções verificam se uma data (ehDataValida) ou um horário (ehHorarioValido) fornecido é válido

```
int ehDataValida(int dia, int mes);
int ehHorarioValido(int hora, int minuto);
```

Função de Entrada de Evento:

```
void leitura(struct evento *x, int contador)
```

Função de Entrada de Evento: Aceita a entrada do usuário para popular os detalhes de um evento. Garante entrada válida para data e horário.

Funções de Exibição:

```
void mostrar_um(struct evento x);
void mostrar_tudo(struct evento *x, int contador);
```

void mostrarAmbientes(struct ambiente *ambientes, int numAmbientes);

Funções de Exibição: Exibem detalhes de um único evento (mostrar_um), de todos os eventos (mostrar_tudo) e de ambientes/recursos associados a eventos (mostrarAmbientes).

Função de Adição de Ambiente:

void adicionarAmbiente(struct ambiente *ambientes, int *numAmbientes, struct evento *eventos, int numEventos);

Função de Adição de Ambiente: Permite ao usuário adicionar um ambiente/recurso e associá-lo a um evento existente.

Função de Remoção de Evento:

int remover(struct evento *x, int contador);

Função de Remoção de Evento: Remove um evento com base na entrada do usuário (data e horário de início).

Função Principal:

int main() {

// Implementação da função principal

}

Função Principal: O corpo principal do programa, que integra as funções definidas para criar uma interface de usuário e gerenciar eventos e ambientes.

Configuração Inicial:

- O programa inicia configurando a localidade para Português e declara variáveis essenciais.

Seleção do Tipo de Usuário:

- O usuário escolhe entre os perfis de Administrador, Professor ou Aluno.
- O código executa diferentes blocos de acordo com o perfil selecionado.

-Perfil do Administrador:

- Apresenta um menu específico para o Administrador com opções como adicionar ambientes, cadastrar eventos, mostrar detalhes, remover eventos, etc.

- As funções `adicionarAmbiente` e `remover` são chamadas conforme necessário.

Perfil do Professor:

- Um menu específico para o Professor é apresentado, permitindo o cadastro de eventos e a visualização de todos os eventos.

Perfil do Aluno:

- Apresenta um menu específico para o Aluno, permitindo a visualização de eventos por data ou descrição.

Funções Importantes:

- `adicionarAmbiente`: Adiciona ambientes associados a eventos.

- `remover`: Remove eventos com base em data e hora de início.

- `mostrarAmbientes`: Exibe todos os ambientes e recursos registrados.

- `mostrar_um`: Exibe detalhes de um evento.

Finalização do Programa:

- Libera a memória alocada e encerra o programa.

Esse resumo destaca as principais funcionalidades e estrutura do código.