Lista de Exercícios 2 - Estruturas de Dados Lineares

Oficina de AEDs2 - Prof. Matheus Pereira

1. Exercício 1 (em C)

Contexto: Um sistema embarcado crítico para controle de uma fábrica automotiva precisa gerenciar três tipos de tarefas com diferentes prioridades e comportamentos:

1. **Tarefas de Tempo Real (Alta Prioridade)**: Devem ser processadas imediatamente (sistema de airbags, freios)
2. **Tarefas Periódicas (Média Prioridade)**: Executadas em intervalos regulares (monitoramento de sensores)
3. **Tarefas de Background (Baixa Prioridade)**: Processadas quando o sistema está ocioso (log de dados)

Enunciado: Implemente um sistema integrado que utilize **três estruturas de dados diferentes** para gerenciar esses tipos de tarefas (implemente tratamento de overflow/underflow para todas as estruturas).

O usuário deverá interagir com um menu interativo pelo terminal, sendo capaz de escolher qual ação será tomada (por exemplo: criar tarefa, processar tarefa, promover tarefa) com base em opções apresentadas na tela.

Estruturas Requeridas:

1. **Pilha Estática** (Para tarefas de tempo real)
   - Implemente com array de tamanho fixo
   - Operações: `empilharEmergencia()`, `desempilharEmergencia()`
2. **Fila Circular** (Para tarefas periódicas)
   - Implemente com array e índices circulares
   - Operações: `enfileirarPeriodica()`, `desenfileirarPeriodica()`
3. **Lista Ordenada** (Para tarefas de background)
   - Mantenha ordenada por prioridade (0 = mais alta, 10 = mais baixa)
   - Operações: `inserirBackground()`, `removerBackground()`

Seu código não está restrito somente às funções sugeridas!

Requisitos do Sistema: Utilize a estrutura abaixo para definir as tarefas simuladas.

```c
#define MAX_EMERGENCIA 5
#define MAX_PERIODICA 5
#define MAX_BACKGROUND 30

typedef struct {
    int id ;
    int prioridade ; // 0-10 (0 = maxima prioridade)
} Tarefa ;

typedef struct {
    Tarefa pilha[MAX_EMERGENCIA] ; // pilha de tarefas
    int n; // numero de tarefas na lista
} PilhaEmergencia ;

typedef struct {
    Tarefa fila[MAX_PERIODICA] ; // fila de tarefas
    int primeiro; // posicao do primeiro
    int ultimo; // posicao do ultimo
} FilaPeriodica ;

typedef struct {
    Tarefa lista[MAX_BACKGROUND] ; // lista de tarefas
    int n; // numero de tarefas na lista
} ListaBackground ;
```

Funcionalidades Essenciais para Implementar:

1. **Scheduler Integrado**:

```c
    Tarefa processarTarefa(PilhaEmergencia* p,
                           FilaPeriodica* f,
                           ListaBackground* l);

```

   Que retorna a próxima tarefa a ser executada seguindo as regras:

   - Sempre verifica primeiro a pilha de emergência
   - Depois a fila circular de tarefas periódicas
   - Finalmente a lista de tarefas de background

2. **Mecanismo de Promoção**:

```c
    void promoverTarefa(PilhaEmergencia* p,
                        ListaBackground* l, int id);

```

   Que move uma tarefa da lista de background para a pilha de emergência

3. **Estatísticas do Sistema**:

```c
    void imprimirEstruturas(PilhaEmergencia* p,
                            FilaPeriodica* f,
                            ListaBackground* l);

```

   Que mostra o estado atual de todas as estruturas

Análise de Complexidade Requerida: Para cada operação implementada, analise e justifique a complexidade de tempo das operações individuais de cada estrutura.

Observações

1. Mantenha a ordenação eficiente na lista de background.
2. A prioridade das tarefas só é usada para a ordenação da lista de tarefas de background. **O que define se uma tarefa vai para a pilha, lista ou fila é a entrada do usuário**.