Trabalho 2 - TAD, Projetos com listas, pilhas, filas e deques

Ciência da Computação - Estrturas de Dados I - Prof. Edkallenn

INSTRUÇÕES PARA A RESOLUÇÃO DOS EXERCÍCIOS:

- HÁ TRÊS FORMAS DE ENVIAR O EXERCÍCIO (VOCÊ ESCOLHE UMA DELAS!):

    1. Fazer o upload de cada arquivo-fonte (arquivo com a extensão `.c`) na área de envio do BlackBoard
    2. Criar um *repositório no Github para cada projeto* chamados, cada um, individualmente: `Lista2 - Estruturas de Dados - Prof. Edkallenn - NomeDoProjeto` e fazer o push dos arquivos-fonte (somente os arquivos `.c`) sem os binários e, de preferência com este arquivo (das questões) incluído. **Você envia, portanto, SOMENTE O LINK DO REPOSITÓRIO.** O repositório não pode ter sido clonado sob pena de **ANULAÇÃO DO EXERCÍCIO.**
    3. Criar um **Repl** (repositório de projeto no Replit.com) e compartilhar o link com os arquivos do exercício (os arquivos-fonte `.c`). **Você envia, somente o link**.
    - TODOS os arquivos-fonte tem uma forma de serem nomeados. Veja abaixo como cada arquivo deve ser nomeado.

- Os programas **NÃO DEVEM SER COMPACTADOS**. O código-fonte deve ser enviado via upload diretamente na resposta do exercício (arquivo por aquivo) conforme acima.

- **Cada arquivo deve ter o seguinte formato: `ED-lista2-questaoXX` onde `XX` é o número da questão correspondente. Em caso de TADs com mais de um arquivo, os demais seguem a mesma regra, porém com a extensão `.c` ou `.h`**

- **IMPORTANTE: NÃO SERÃO ACEITOS TRABALHOS QUE NÃO ESTIVEREM NO FORMATO ACIMA**

- **OBSERVAÇÃO**: **TODOS** os programas entregues devem ter o seguinte cabeçalho:

```
/*
**    Função :
**    Autor :
**    Data  :
**    Observações:
*/
```

Onde deverá estar escrito o que o programa faz, o autor (nome, turma, a data e as observações que forem pertinentes.

Os trabalhos não serão aceitos após a data SOB HIPÓTESE ALGUMA.

---

1. TAD Vetor de Inteiros

Implemente um TAD para representar dinamicamente um vetor de números inteiros e que ofereça as seguintes operações: criação do vetor, inserção de um elemento, remoção de um elemento, consulta do elemento em uma determinada posição, impressão dos elementos do vetor e remoção do vetor inteiro. Este TAD obrigatoriamente terá que usar 2 arquivos (a interface e a implementação) e mais um arquivo de teste.

2. Sistema de Agendamento de Consultas:

Criar um Sistema de agendamento de consultas para uma clinica médica.

Na rotina da clínica existem 2 tipos de agendamento:

- Agendamento de consultas com Médico. O médico atende até 5 pessoas por dia.
- Agendamento de consultas com Enfermagem.
- A Enfermagem atende 24 horas e sem limite de agendamento.

Para isso será necessário a utilização de 2 estruturas de dados:

- 1 Fila Estática para agendamento com Médico. Esta estrutura permite o controle do limite de agendamentos, atraves da implementação interna da estrutura.
- 1 Fila Dinâmica para agendamento com Enfermagem. Esta estrutura permitirá agendar consultas quantas consultas forem necessárias, sem limite de pacientes.

3. Sistema de Registro de Estudantes

Objetivo:

Desenvolver um sistema de registro de estudantes em C, utilizando TADs, Listas, Pilhas, Filas e Deques para gerenciar informações dos estudantes.

Funcionalidades:

1. **Definição do TAD Estudante:**
   - Criar um TAD para representar as informações de um estudante, como nome, número de matrícula, notas, etc.
2. **Listas:**
   - Utilizar listas encadeadas para armazenar informações dos estudantes.
   - Permitir adição, remoção e busca de estudantes na lista.
3. **Pilhas:**
   - Implementar uma pilha para armazenar históricos de notas de um estudante.
   - Ao atualizar as notas, empilhar o histórico anterior.
4. **Filas:**
   - Utilizar uma fila para simular uma fila de espera de estudantes para matrícula em disciplinas.
   - Adicionar estudantes à fila e remover quando matriculados.
5. **Deques:**
   - Utilizar um deque para gerenciar uma lista de espera para atendimento.
   - Adicionar estudantes no início ou final da fila de espera.
6. **Funções adicionais:**
   - Implementar funções para calcular médias de notas de um estudante ou percorrer estruturas de dados.

Exemplo de Estrutura do TAD:

```
typedef struct {
    char nome[50];
    int matricula;
    float notas[3];  // Supondo 3 notas por estudante
} Estudante;
```

Fluxo Geral do Programa:

1. **Inicialização:**

    - Criar uma lista de estudantes vazia, uma pilha para históricos de notas, uma fila para matrículas e um deque para atendimentos.
2. **Adição de Estudantes:**
    - Permitir ao usuário adicionar novos estudantes à lista.
3. **Atualização de Notas:**
    - Ao atualizar as notas de um estudante, empilhar o histórico anterior, por disciplina.
4. **Matrícula em Disciplinas:**
    - Adicionar estudantes a uma fila de espera para matrícula em disciplinas.
    - Remover estudantes da fila quando matriculados.
5. **Atendimento:**
    - Utilizar um deque para gerenciar uma lista de espera para atendimento.
    - Adicionar estudantes no início ou final da fila de espera.
6. **Funções:**
    - Implementar funções para calcular médias de notas ou realizar operações específicas nos dados dos estudantes.

Observações:

- Este projeto oferece uma abordagem prática para manipulação de dados de estudantes usando várias estruturas de dados.
- Explorem a eficiência das operações de inserção, remoção e busca em listas e deques, considerando os contextos específicos.
- Caso seja interessante, acrescentem uma opção para salvar os dados dos alunos, disciplinas e notas em arquivos.
- Experimentem as diferentes estruturas de dados e a implementem suas próprias funções recursivas ou não para aprofundar a compreensão dos conceitos.

4. Sistema de Gerenciamento de Tarefas (To-Do List)

Objetivo:

Desenvolver um sistema de gerenciamento de tarefas (To-Do List) em C, aplicando diferentes estruturas de dados para armazenar e manipular as tarefas.

Funcionalidades:

1. **Definição do TAD:**
    - Criar um TAD para representar uma tarefa.

2. **Listas:**
    - Utilizar uma lista encadeada para armazenar as tarefas a serem realizadas (To-Do List).
    - Adicionar tarefas no início ou no final da lista.
3. **Pilhas:**
    - Implementar uma pilha para armazenar tarefas concluídas.
    - Quando uma tarefa é concluída, ela é removida da lista e movida para a pilha de tarefas concluídas.
4. **Filas:**
    - Utilizar uma fila para agendar tarefas com data de vencimento.
    - Ao adicionar uma tarefa com uma data futura, ela é colocada na fila para ser movida para a To-Do List quando a data chegar.
5. **Deques:**
    - Implementar um deque para permitir a remoção eficiente tanto do início quanto do final da lista de tarefas.
6. **Funções adicionais:**
    - Implementar uma função recursiva para percorrer a lista de tarefas e realizar alguma operação específica (ex: contar o número total de tarefas).

Exemplo de Estrutura do TAD:

```
typedef struct {
    char descricao[100];
    int prioridade;  // 1 (baixa), 2 (média), 3 (alta)
    int dataVencimento;  // formato DDMMYYYY
} Tarefa;
```

Fluxo Geral do Programa:

1. **Inicialização:**
    - Criar listas, pilhas e filas vazias.
2. **Adição de Tarefas:**
    - Permitir ao usuário adicionar tarefas à To-Do List, especificando descrição, prioridade e, opcionalmente, data de vencimento.
3. **Conclusão de Tarefas:**
    - Ao concluir uma tarefa, removê-la da To-Do List e movê-la para a pilha de tarefas concluídas.
4. **Agendamento de Tarefas:**
    - Permitir ao usuário agendar tarefas para uma data futura.

5. **Recursividade:**
    - Implementar uma função recursiva para percorrer a lista de tarefas e realizar uma operação específica.

Observações:

- Caso seja interessante, tentem implementar o salvamento das tarefas em disco
- Eu sugiro começarem a trabalhar usando gerenciadores de projeto reais, como o Trello.

  Usem como modelo o quadro que está disponível no link abaixo:
- Ao acessar o link acima, COPIEM o quadro para a própria conta e editem normalmente.

👌 Bom Exercício! 😘

DICA: É IMPORTANTE QUE TODOS OS PROJETOS SEJAM REALIZADOS! E QUE VOCÊ MESMO FAÇA OS EXERCÍCIOS. SE NÃO CONEGUIR FAZER TODOS, ENVIE TODOS AQUELES QUE VOCÊ CONSEGUIR. NÃO DEIXE DE ENVIAR O TRABALHO. SUA NOTA DEPENDE ELE.

🏫Prof. Ed.