

Desafio: Tipo Abstrato de Dados



FILA FIFO

Uma organização está construindo uma aplicação que recebe milhares de requisições para atendimento de saúde. Essas requisições possuem as seguintes informações:

- Nome do Paciente – uma *string* de 40 caracteres
- Código de inscrição que o identifica no sistema de saúde – um número de tipo inteiro
- Código do procedimento solicitado – uma *string* de 10 caracteres

Essas requisições precisam ser organizadas por ordem de chegada e são consumidas pela aplicação na medida que se consegue alocar o paciente na instituição que irá atendê-lo. Assim, é necessário a criação de um **TAD** (Tipo abstrato de dados) que permita a inserção desta requisição, que será realizada quando ela chegar, e permita a remoção dela quando a aplicação conseguir alocá-la em uma instituição.

Outra característica necessária do TAD é fornecer a quantidade de requisições de espera, pois, conforme o tamanho, as instituições parceiras da organização também podem ser usadas para acelerar o processo de atendimento dos pacientes.

A construção precisa ser em linguagem C e, ao final de sua elaboração, devem constar dois arquivos:

- O cabeçalho do TAD denominado `estrutura.h`
- A implementação do TAD com o código-fonte, denominado `estrutura.c`, e o compilado, `estrutura.o`.

A estrutura da requisição é fornecida pelos arquivos:

- `requisicao.h`
- `requisicao.o`

Para seus testes de validação de sua implementação, está sendo fornecido um exemplo de código com uma função `main`, que pode ser usada para verificar seu funcionamento: - **`teste.c`**

Este arquivo pode ser baixado assim como o desafio e a agenda. Este será o teste realizado na avaliação de seu código.

Mas, atenção! **Funcionar é um elemento básico para o desafio, mas não é suficiente.** A avaliação do desafio irá considerar os seguintes aspectos:

- O código funciona, apresentando os dados corretos no teste original proposto [`teste.c`].
- O código é otimizado, consumindo o mínimo de memória (não há variáveis desnecessárias, não estão sendo armazenados dados temporários desnecessariamente?).
- O código é otimizado no tempo de resposta de inserção e remoção (O código está com a menor complexidade possível para o caso?).
- O código atende a todos os requisitos demandados.