SCC0202 - Algoritmos e Estruturas de Dados I

Profs. Rudinei e JB

Monitores: Laura e Felipe

Projeto 1: Pronto Socorro SUS

Equipe: máximo 3 pessoas. Não hesite em pedir ajuda. Lembre-se: plágio é uma forma de corrupção.

1 Introdução

O serviço de atendimento médico emergencial envolve várias etapas e obedece a um protocolo: o acidentado dá entrada ao pronto socorro e vai para uma sala de espera. O primeiro a chegar será o primeiro a ser atendido. Para cada paciente que dá entrada, o médico estabelece um "histórico de tratamento" com ações que podem ser desfeitas, a depender do seu estado de saúde. Melhor explicando: um atendimento médico envolve a administração de medicamentos em sequência, que pode ser desfeita: o último medicamento prescrito será o primeiro a ser suspenso. Por fim, o registro do paciente deve permanecer no hospital, mesmo após a sua alta médica. Caso ele volte à emergência, seus dados já estarão registgrados.

2 Objetivo

O seu objetivo é simular um serviço de atendimento médico de pacientes em um ambiente de Pronto Socorro (PS), por meio de **Tipos Abstratos de Dados** tais como: pilhas, listas e filas. Os dados devem ser armazenados de forma persistente, ou seja, tudo que é registrado no sistema deve ser armazenado em disco. Nesse documento, detalharemos o comportamento do sistema, a estrutura e funcionalidades de cada TAD que você deve implementar.

3 O Paciente

É o elemento central do seu sistema. Dele, basta armazenar o nome e um ID (chave única de identificação). Nota: se não explicitamente especificados, os tipos de dados tem implementação livre. Exemplo: ID pode ser um inteiro, "string", etc. Você decide.

4 TADs sugeridos

Os tipos de dados (int, float, string, etc), os "containers" para cada TAD (estático ou dinâmico) e o que cada função retorna ficam a seu critério. Abaixo, daremos dicas das informações (dados) e funcionalidade (funções utilitárias) que cada TAD deve conter.

4.1 Relação de Pacientes

O hospital precisa manter uma lista de pacientes atendidos no PS. A lista deve conter as seguintes informações: o paciente e o seu histórico de tratamento.

As funcionalidades para a lista são: inserir paciente, apagar paciente, buscar paciente. Além disso deve ser possível listar todos os pacientes.

Nota: não há um limite para a quantidade de pacientes registrados no hospital. A rigor, o limite é quantidade de memória do servidor.

4.2 A triagem dos pacientes

Ao dar entrada na emergência, um paciente, já registrado ou não no hospital, vai para uma sala de espera. Por ora, não haverá prioridade no atendimento. O critério é a ordem de chegada. Quem chega primeiro, é atendido primeiro.

As funcionalidades: inserir paciente, remover paciente. Nota: o hospital tem uma capacidade limitada de atendimento na emergência, ou seja, a fila tem tamanho finito. Determine este tamanho e implemente tb as funcionalidades: fila cheia e fila vazia.

4.3 O histórico médico do paciente

Como já mencionado,o histórico de atendimento de um paciente no seguinte esquema: o último procedimento sugerido pode ser desfeito, ou seja, ele será o primeiro a ser descartado (LIFO). A quantidade de procedimentos para cada paciente é de, no máximo, 10. Um histórico é simplesmente um texto de, no máximo 100 caracteres.

Funcionalidades: inserir, retirar, consultar. Analogamente à fila, verificar também: histórico cheio e histórico vazio.

5 A interface do Sistema

Pense que na portaria do PS existirá um atendente operando o sistema e um monitor na sala de espera visível aos pacientes. Vamos sugerir um seguinte menu principal:

1. Registrar paciente
2. Registrar óbito de paciente
3. Adicionar procedimento ao histórico médico
4. Desfazer procedimento do histórico médico
5. Chamar paciente para atendimento
6. Mostrar fila de espera
7. Mostrar histórico do paciente
8. Sair

5.1 Registrar paciente

Se for a primeira ocorrência, colocar paciente na lista e, obviamente, inseri-lo na fila de espera. Nota: não são aceitos IDs repetidos. Sua interface deve ser amigável e reportar isso.

5.2 Registrar óbito de paciente

Em caso de morte do paciente (isso acontece em serviços de emergência, infelizmente), por conta da LGPDb (Lei Geral de Proteção de Dados, versão beta), o paciente deve ser "esquecido" pelo sistema hospitalar. Isso significa que o registro permanente dele no hospital deve ser apagado.

Nota: nosso serviço de emergência é misterioso: um paciente só morre se já tiver sido chamado para atendimento (veja opção 5.4). Caso ainda esteja na fila de espera, é proibido morrer!

Novamente, pensando na interface amigável, reporte ao usuário do sistema o sucesso ou insucesso da transação.

5.3 Adicionar/Desfazer procedimento do histórico médico

Adicionar: pelo ID do paciente, localizar o registro e inserir o item. Faça o procedimento análogo para desfazer, nesta caso informando qual foi o procedimento desfeito. Nota: novamente a interface deve ser amigável: se paciente não for encontrado ou se não houver procedimento a desfazer, reportar.

5.4 Chamar paciente para atendimento

Tirar o paciente da fila! Nota: se a fila estiver vazia, reportar.

5.5 Mostrar fila de espera

Não requer comentários

5.6 Mostrar histórico do paciente

Não requer comentários

6 Persistência dos dados

Persistência dos dados é manter armazenar todas as informações em disco. Não é preciso persistir cada operação separadamente, *on-the-fly*. Por simplicidade, sugerimos o seguinte comportamento: ao sair do sistema, armazene tudo de uma única vez (a lista de paciente, a fila de pacientes e os históricos de atendimento). Ao entrar no sistema, recarregue tudo.

Nota: Isso pode, e deve, ser feito com um TAD! Sugestão de nome: IO (Input/Output). Para facilitar, nós (Rudinei e JB) criaremos um protótipo para você usar e deixaremos disponível para você adaptar ao seu uso. Mas isso você também já viu em ICCI !

Outra opção é vc estender os TADs já existentes e implementar funções utilitárias do tipo *save()* e *load()*. Cada um de seus TAds pilha, fila e lista teriam um par de métodos que escreveriam para o disco e leriam do disco.

7 Linguagem e boas práticas de codificação

Você pode usar C ou C++. Não se esqueça da separação entre interface e implementação: defina a interface do seu TAD em um arquivo .h e a implementação em um arquivo .c (ou .cpp).

Nota: caso opte por C++, lembre-se que não é pra usar class. Mantenha a implementação com struct, como usado em C.

8 Momento Filosófico/Cultural!

Muito provavelmente, se você colocar esta especificação no chatGPT ou papagaio estatístico semelhante, terá o trabalho feito praticamente sem esforço algum. Aí eu lhe pergunto (no sentido filosófico da palavra): você é um ser humano ou um rato?!!

Como sugerimos em sala de aula: usar chatGPT é legítimo, mas no momento certo. Nada substitui a sua dedicação pessoal no início, na definição dos TADs e funcionalidades. Consulte o chatGPT a posteriori, se necessário, depois de ter feito a sua implementação inicial, para lhe ajudar a corrigir erros. Vocês estão no primeiro ano, período em que é essencial desenvolver habilidades cognitivas para a construção de algoritmos e estrutura de dados. E isso se consegue usando o seu cérebro, e não o do chatGPT.