SCC0202 - Algoritmos e Estruturas de Dados I Profs. Rudinei e JB

Mnitores: Laura e Felipe

Projeto 2: Pronto Socorro SUS - V2

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 virou uma iniciativa nacional e agora vai atender centenas de milhares de pacientes. Isso implica que as inserções, alterações e buscas no cadastro de acidentados atendidos deve ser muito eficiente, não sendo mais aceitas operações lineares, já que os requisitos do sistema demandam um tempo de resposta muito rápido.

Mas desta feita, o serviço de emergência foi alterado e agora o atendimento não é por ordem de chegada. O atendimento obedece uma prioridade, assim definida: 1) Emergência: caso gravíssimo com risco de morte; 2) Muito urgente: grave e risco de evoluir para morte; 3) Urgente: gravidade moderada e necessidade de atendimento médico, sem risco imediato; 4) Pouco Urgente: poderia ser atendido numa Unidade Básica; 5) Não Urgência: risco algum (resfriados, espinho no pé, etc!)

Portanto, quando um paciente dá entrada, será feita uma análise do seu quadro clínico que o classificará em uma das 5 categorias acima. Caso mais de um paciente estejam na mesma categoria, adota-se a ordem de chegada na emergência.

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**. De forma que 2 requisitos sejam satisfeitos: melhor eficiência computacional de processamento e uso estritamente necessário de memória, já que os servidores não são muito poderosos. Os dados devem ser armazenados de forma persistente, ou seja, tudo que é registrado no sistema deve ser armazenado em disco.

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 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. Remover paciente
3. Listar pacientes
4. Buscar paciente por ID
5. Mostrar fila de espera
6. Dar alta ao paciente.
7. Sair

4.1 Registrar/Remover/Listar pacientes

Se for a primeira ocorrência, cadastrar o paciente no hospital e, obviamente, colocá-lo em espera, levando-se em conta as prioridades acima estabelecidas. Nota: não são aceitos IDs repetidos. Sua interface deve ser amigável e reportar isso.

Remoção de paciente dos registros do hospital. Isso só será possível se este não estiver aguardando atendimento.

Listar pacientes: mostrar todos os pacientes do hospital.

4.2 Buscar paciente por ID

Não requer comentários

4.3 Mostrar fila de espera

Listar os pacientes da MAIOR para a MENOR prioridade. Pacientes com a mesma prioridade devem ser listados de acordo com a ordem de chegada na emergência.

4.4 Dar alta ao paciente

O paciente foi atendido e deixou o serviço de emergência. Ele permanece nos registros do hospital.

5 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.

6 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).