Instituto Federal do Espírito Santo — IFES (Campus Serra)
Coordenadoria de Informática — Cin

Trabalho de Programação

Simulação de Realização de Exames de Raio-X (parte 2)

Valor: 30 pontos
Deadline: 22 de agosto de 2024

Prof. Thiago M. Paixão
thiago.paixao@ifes.edu.br

1 Objetivo

O trabalho prático de programação consiste em simular o processo de realização de exames de raio-X de tórax e diagnóstico em um hospital. A ênfase do trabalho está na organização de filas em diversas etapas do processo. A cada momento, pacientes chegam ao hospital e exames são realizados conforme a disponibilidade dos aparelhos. A IA[1] sugere diagnósticos preliminares, e os exames são encaminhados para laudo conforme a disponibilidade do médico responsável.

Nesta etapa do trabalho, você implementará o processo de simulação com base nos TADs desenvolvidos para a entrega 1. TADs adicionais serão necessários para a implementação das filas de espera.

As principais competências a serem desenvolvidas neste trabalho incluem:

- Implementação de um sistema de simulação.
- Uso e implementação de filas.
- Implementação de módulos e TADs.
- Manipulação de arquivos.
- Documentação da solução.

2 Simulação de exames de Raio-X e diagnóstico

A Figura 1 fornece uma visão geral do processo de simulação. Pacientes chegam ao hospital e são organizados em uma fila "virtual" de atendimento (`PatientQueue`). Os registros dos pacientes são armazenados em um "banco de dados" implementado como um simples arquivo texto (`db_patient.txt` ↪). À medida que uma máquina fica desocupada, o gerenciador das máquinas (`XRMachineManager`) aloca um paciente (o primeiro da fila) para realização do exame. Após finalização, é gerado um registro com os dados do exame. A IA "analisa" o exame e atribui uma pré-condição/patologia (`condition_IA`) ao exame. O registro com a informação da patologia é gravado em um banco de dados (`db_exam.txt`).

Os exames são alocados numa fila de espera para laudo de acordo com a gravidade da patologia (fila de prioridades `ExamPriorityQueue)`. Ou seja, exames cuja patologia apresentar prioridade mais elevada devem ser analisados antes, independentemente da ordem de chegada. O médico (que trabalha numa escala 24/7, não dorme, não se alimenta e nem toma banho) é responsável por analisar o primeiro exame da fila (de maior prioridade) e elaborar o laudo (`Report`) confirmando (ou corrigindo) o diagnóstico fornecido pela IA. O laudo produzido deve ser armazenado em um banco de dados implementado em arquivo texto (`db_report.txt`).

[1] Não implementaremos a IA propriamente dita, apenas simularemos seu resultado.

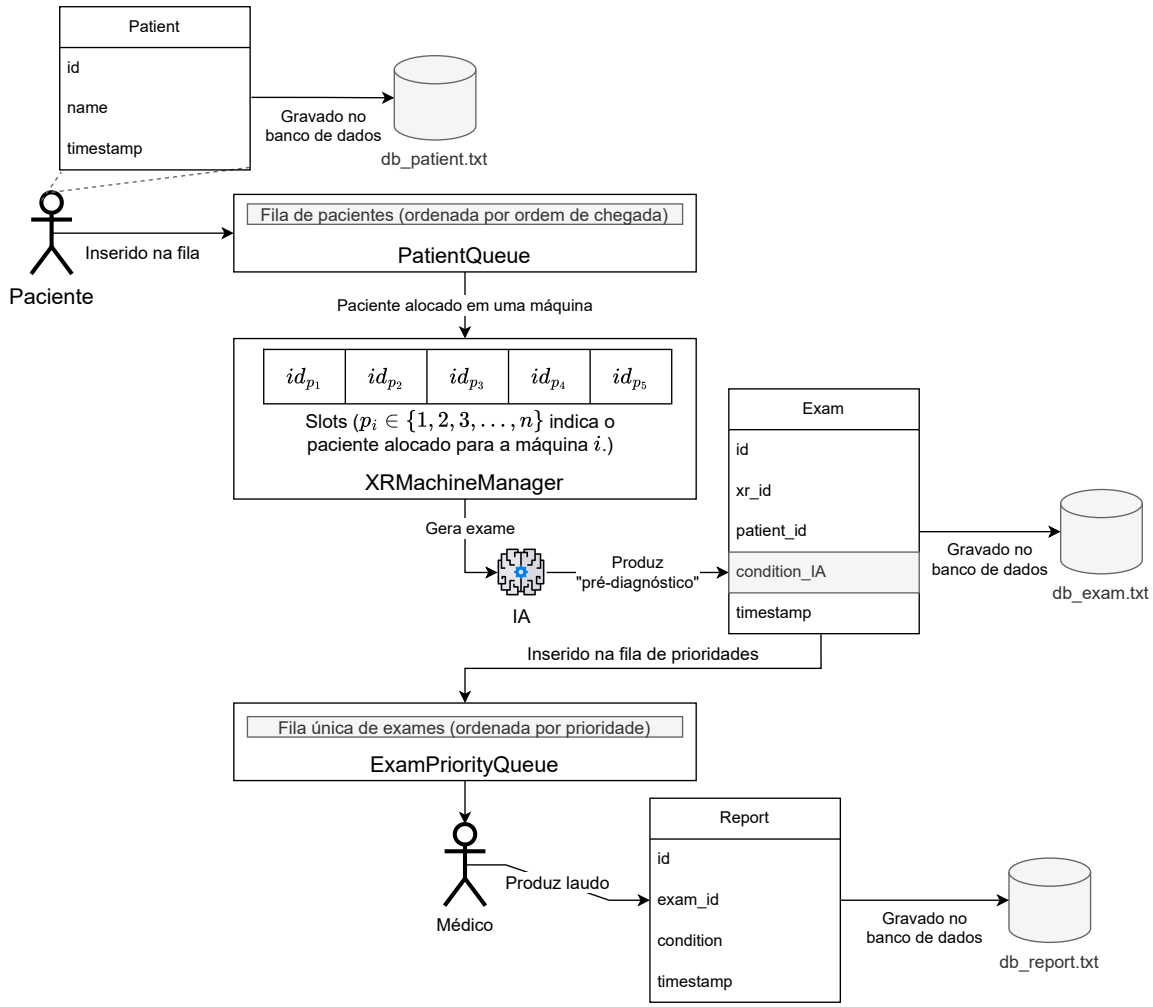

Figura 1: Visão geral do processo de simulação e principais estruturas de dados.

O processo de simulação é baseado em eventos (ex. chegada de um paciente e finalização de um exame) que ocorrem a cada instante de tempo com uma certa probabilidade. No total, a simulação deve durar 43.200 unidades de tempo. Os detalhes do processo de simulação são fornecidos a seguir.

2.1 Chegada de pacientes

A cada instante de tempo, no máximo 1 (um) paciente pode chegar ao hospital. A probabilidade deste evento ocorrer é 20%. Para cada novo paciente, é criado um registro com identificador (`id`), nome e o instante de chegada no hospital (`timestamp`). Esse registro é inserido numa *fila de pacientes* e seus dados são gravados num arquivo texto (`db_patient.txt`) simulando um banco de dados.

2.2 Exame e pré-diagnóstico

Havendo aparelho disponível (de um total de 5), o primeiro paciente da fila de pacientes é alocado para realização do exame no aparelho em questão. A duração do exame é fixo: 10 unidades de tempo. Após finalizado, é gerado o registro de exame (`Exam`). Assumimos que aparelhos só ficam ociosos se a fila de pacientes estiver vazia.

O exame recém-realizado é analisado pela IA, conforme Figura 1, a fim de produzir um "pré-diagnóstico", isso é, determina a condição (normal ou patológica) do paciente. Neste trabalho, a operação da IA

Condição	Probabilidade de Ocorrência	Nível de Gravidade
Saúde Normal	0.3	1
Bronquite	0.2	2
Pneumonia	0.1	3
COVID	0.1	4
Embolia pulmonar	0.05	4
Derrame pleural	0.05	4
Fibrose pulmonar	0.05	5
Tuberculose	0.05	5
Câncer de pulmão	0.1	6

Tabela 1: Lista de patologias com probabilidades de ocorrência e níveis de gravidade.

é simulada, sendo a condição do paciente determinada aleatoriamente com base na distribuição de probabilidades na Tabela 1.

Uma vez determinada, a condição do paciente detectada no exame pela IA é gravada no registro do exame (`condition_IA`), que, por sua vez, é gravado em disco (`db_exam.txt`).

2.3 Fila de prioridades

A partir da condição fornecida pela IA, o exame pode ser categorizado em um dos níveis de gravidade apresentados na Tabela 1. Essa informação é utilizada para organizar os exames em uma fila de prioridade (`ExamPriorityQueue`), garantindo que casos de maior gravidade sejam analisados com antecedência.

2.4 Realização de laudos

A realização de um laudo (`Report`) consiste em remover exames da fila de prioridades e determinar a condição do paciente (`condition`). Isso é feito de forma simulada baseado no seguinte algoritmo:

1. Gere um valor de probabilidade $p \in [0, 1]$ baseado em distribuição uniforme.
2. Se $p < 0.8$, mantenha a condição fornecida pela IA;
3. Caso contrário, faça:
   (a) Determine aleatoriamente uma nova condição seguindo a distribuição de probabilidades da Tabela 1.
   (b) Se a condição determinada for diferente daquela fornecida pela IA, assinale tal condição ao laudo.
   (c) Caso contrário, volte ao passo 3a.

Após geração do laudo, o mesmo deve ser gravado em banco de dados (`db_report.txt`).

3 Relatório de simulação

Periodicamente, seu programa deve imprimir um relatório de simulação com as informações descritas a seguir.

- Número de pacientes que visitaram o hospital.
- Número pacientes na fila aguardando exame.

- Número de pacientes que já realizaram exame e, dentre estes, a porcentagem do que já receberam laudo.
- Tempo médio de laudo: tempo médio que os exames ocupam a fila de prioridades.
- Tempo médio de laudo por condição: tempo médio que os exames de uma condição específica (assinalada pelo médico) ocupam a fila de prioridades.
- Número de exames realizados após o limite de tempo estabelecido (7.200 unidades de tempo).

4 Requisitos do programa

Neste trabalho, você terá a flexibilidade de implementar/adequard os módulos e os Tipos Abstratos de Dados (TADs) que considerar necessários para a simulação. No entanto, é fundamental que o arquivo `main.c` contenha a implementação da lógica principal da simulação, o que inclui o *loop* de contagem de tempo principal. A cada n unidades de tempo (n será escolhido por você), o relatório de simulação deve ser atualizado na tela. Utilize alguma função de espera (ex. `sleep()`) para controlar a velocidade de execução e exibição das informações.

5 Critérios de avaliação

A avaliação deste trabalho levará em consideração os seguintes critérios:

1. Implementação e uso adequado de estruturas de dados: Até **15 pontos** serão atribuídos à implementação correta das estruturas de dados necessárias para a simulação, como filas e listas, bem como ao seu uso adequado durante a simulação.
2. Lógica e organização do código: Até **7,5 pontos** serão concedidos pela clareza, organização e boas práticas de codificação no projeto. Isso inclui a estruturação adequada do código (modularização), nomes significativos para variáveis e funções, e formatação consistente.
3. Implementação das métricas: Até **3 pontos** serão atribuídos à implementação correta das métricas de desempenho especificadas no trabalho.
4. Documentação do código: Até **1,5 pontos** serão atribuídos à qualidade da documentação incorporada ao código. Certifique-se de incluir comentários significativos que expliquem a lógica por trás das implementações.
5. README.md descritivo: Até **3 pontos** será concedido pela criação de um arquivo README.md descritivo e informativo no seu repositório. O README deve fornecer informações claras sobre como executar e utilizar o seu projeto. Além disso, deve exibir a estrutura do projeto, apresentar as principais estruturas de dados e mencionar as principais decisões de implementação.
6. Apresentação (a ser agendada): Um fator real α será aplicado após a apresentação do projeto. α será um número entre 0 e 1, refletindo a qualidade da apresentação e a capacidade de explicar e defender o projeto.
7. Correção: Será atribuído $\beta = 1$ se não houver falhas críticas no projeto; caso contrário, $\beta = 0$. Isso avaliará se o projeto funciona conforme o esperado e se atende aos requisitos especificados.
8. Dias de atraso: Será descontado 20% do valor total do trabalho a cada dia de atraso.

A nota será calculada utilizando a equação $nota = (1 - d/5) \times \alpha \times \beta \times P$, onde P é a soma dos pontos dos critérios 1 a 4. É importante notar que a nota será zerada após 5 dias de atraso.

Importante: O programa será testado num ambiente Linux Ubuntu 22.04 com GCC 11. Recomendo FORTEMENTE desenvolver e testar nesse ambiente.

6 O que entregar?

1. Um link para um repositório .git com código fonte do projeto: `Makefile` e arquivos `.c` e `.h`.
2. A documentação/relatório será feita no arquivo README.md do repositório e deverá explicar o passo-a-passo para executar o programa, os principais TADs e as principais decisões de implementação.

Bom trabalho!