IFMG - Campus São João Evangelista
Sistemas de Informação
Turma: SI 221 2022-2

Professor: Eduardo Trindade
eduardo.trindade@ifmg.edu.br
Algoritmos e Estrutura de Dados I

Trabalho Prático III - 10 pontos

Observações

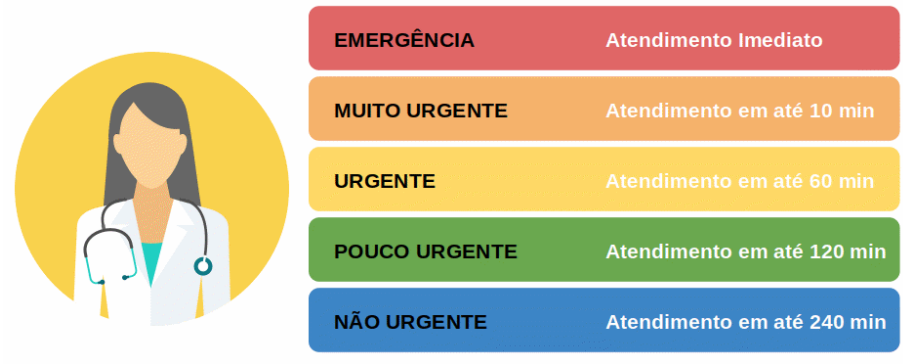
- O trabalho é **individual**.
- É recomendado discutir os problemas e estratégias de solução com seus colegas.
- Tente solucionar os problemas vocês mesmos, pois solucionar problemas e desenvolver o raciocínio lógico é componente fundamental neste curso.
- A entrega deverá ser feita **exclusivamente** pelo Moodle.
- **Atenção:** Entregar em um único arquivo .zip, (ou .rar ou .7z ou .tar.gz) contendo o(s) arquivo(s) necessário(s) para a compilação de seu projeto (exemplo: main.cpp). O nome do arquivo .zip deve ser TP3AEDS1SeuNome.zip.
- A data limite de entrega encontra-se no Moodle.

Objetivos

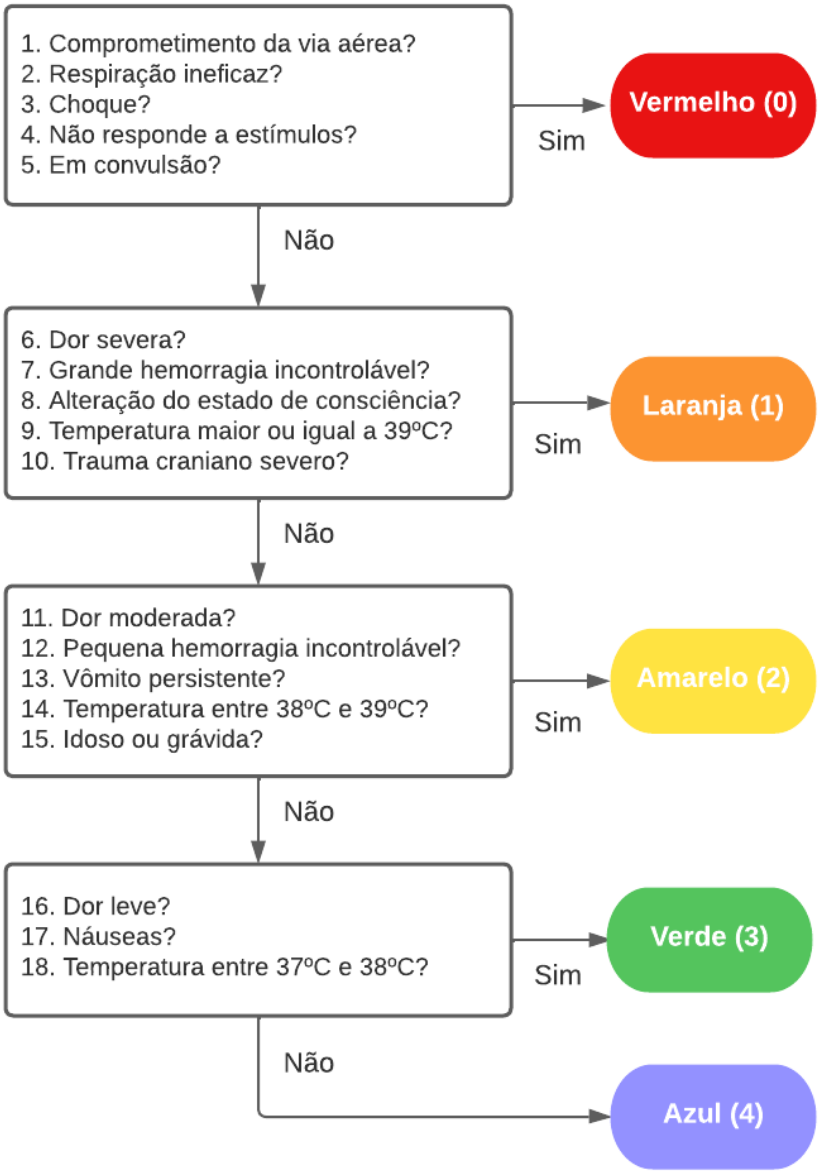
- Fixar conceitos sobre estruturas de dados e modularização.
- Trabalhar com TAD de Fila com Ponteiros e suas variações.
- Estimular o raciocínio lógico e a proposição para solução de problemas.

Descrição

As Unidade de Pronto Atendimento (UPA) funcionam 24 horas por dia, sete dias por semana, e podem atender grande parte das urgências e emergências. Neste trabalho simularemos os procedimentos de classificação de risco para definir prioridades nos atendimentos a pacientes (filas de espera). A prioridade é definida pelas cores vermelho (0), laranja (1), amarelo (2), verde (3) e azul (4), cada qual com um determinado tempo de espera tolerável, chamados Protocolo de Manchester. A Figura a seguir resume cada caso.



Um exemplo de protocolo para encontrar a prioridade é definido pelo diagrama a seguir:



Tarefa

Você deverá implementar um programa em C/C++ que controle a fila de pacientes a serem atendidos e os pacientes em atendimento médico. Deverão ser implementadas as seguintes funcionalidades:

1.) **Registro de Hospital (UPA)**: Uma estrutura responsável por armazenar uma lista de médicos. Providencie variáveis para guardar a quantidade de médicos cadastrados, quantidade de pacientes na UPA, quantidade de médicos disponíveis, total de pacientes atendidos, etc. Essa estrutura não será cadastrada como entrada pelo usuário, mas atualizada mediante as ações no sistema.

2.) **Cadastro de Médico**: Uma estrutura simples responsável por armazenar dados cadastrais referentes aos médicos (Nome, especialidade, CRM).

3.) **Triagem:** Essa estrutura deverá armazenar a resposta das 18 questões propostas no Protocolo de Manchester. Obviamente, se for detectado prioridade de Emergência, por exemplo, dispensa-se as análises restantes.

4.) **Cadastro de Paciente:** Uma estrutura para armazenar os dados principais de um paciente (nome, endereço, idade, sexo). Além disso, a estrutra deverá estar relacionada à estrutura da Triagem e a um inteiro responsável por guardar o valor de sua prioridade. O cadastro de pacientes deverá ser chamado somente ao registrar um Novo Atendimento.

5.) **Novo Atendimento**: Para cada pessoa que chega, uma prioridade será definida e essa pessoa poderá ser alocada a um médico. Cada médico demora um certo tempo (dentro de um intervalo esperado) para atender um paciente, e por meio desse tempo o programa deverá simular o tempo de atendimento e espera de pacientes.

A partir do tempo de espera tolerável de cada cor (descrito no Diagrama anterior) é possível saber se há uma demanda maior do que a capacidade da UPA. Os tempos que os pacientes demoram para ser atendidos são, segundo suas cores:

- 0 - Vermelho: 50 minutos em média, podendo variar em 10 minutos para mais ou menos.
- 1 - Laranja: 20 minutos em média, podendo variar em 5 minutos para mais ou menos.
- 2 - Amarelo: 15 minutos em média, podendo variar em 5 minutos para mais ou menos.
- 3 - Verde: 10 minutos em média, podendo variar em 2 minutos para mais ou menos.
- 4 - Azul: 5 minutos em média, podendo variar em 3 minutos para mais ou menos..

O sistema deve funcionar como chamada para atendimentos. Cada chamada para atendimento é realizada no instante em que um paciente dá entrada no pronto-socorro. Assim, o paciente deverá passar por uma triagem (respondendo às 18 questões do Diagrama) antes de ser alocado a um médico.

Exemplo de triagem:

Paciente: Silva
Triagem: N N N N N N N N N N N N S S N N N N

A triagem deverá ser analisada conforme seu preenchimento, obtendo os seguintes resultados:

(13: S; 14: S) = Vômito persistente e Temperatura entre 38ºC e 39ºC = **Urgente: Amarelo**
Previsão de Atendimento: **em até 60 minutos**.

Ao término da triagem, deve-se perguntar se existem mais pacientes aguardando triagem. Se sim, faça a nova triagem e classificação. Caso não haja, verifique a disponibilidade dos médicos e direcione o atendimento. Direcionar o atendimento consiste em relacionar um médico a um paciente, definindo o exato momento de início do atendimento (utilize a biblioteca <time.h> ou outra se achar necessário).

Quando houver médico disponível, o paciente será alocado para o mesmo. Um novo atendimento consiste no desenfileiramento de uma Fila de Espera. Um atendimento finalizado consiste na liberação de um médico para atendimento e de uma mudança de *status* do paciente para **atendido: *true***.

6.) Saída de dados: Deve-se exibir por completo o total de atendimentos no dia, a quantidade de médicos atendendo, a quantidade de médicos livres, a fila de espera, e todas as informações necessárias para compreensão do sistema. Você terá a liberdade para exibir estes dados como achar mais adequado.

Implementação

A implementação deve seguir as seguintes regras:

(1) Todos os dados referentes aos atendimentos devem ser armazenados em *structs* ou *classes* para facilitar o acesso. A estrutura principal do programa deve seguir uma TAD de Fila de Prioridade Dinâmica (Fila com ponteiro).

(2) A UPA comportará até 5 médicos cadastrados.

(3) Deve-se obedecer à seguinte regra: atendimentos menos prioritários, em hipótese alguma serão alocados para atendimento antes dos mais urgentes.

(4) O sistema deverá ser capaz de trabalhar com mais triagens do que número de médicos, a fim de que se possa testar a lista de espera.

(5) Uma função **Atualizar** deverá ser implementada para percorrer todos os atendimentos e verificar se o mesmo já foi finalizado ou não. Um atendimento é finalizado quando atende o tempo de atendimento sorteado ao ser alocado a um médico. Exemplo: um atendimento **Não urgente** (azul) foi alocado no momento X e demorará 4 minutos (240 segundos) para ser finalizado (tempo sorteado aleatoriamente mediante a regra de 5 minutos em média, variando de 3 para mais ou menos). Essa função poderá ser chamada manualmente em um Menu e/ou sempre que uma operaçao de Triagem ou Registro de Atendimento for realizada.

(6) O sistema deve permitir a manipulação das *structs* possibilitando as inclusões, consultas e listagem de maneira eficiente.

(7) Poderão ser utilizadas todas as TAD vistas que auxiliarem na solução do problema.

(8) Deverá ser trabalhada a biblioteca **<time.h>** para registro da chegada do paciente, início do atendimento e duração do atendimento.

Entrega

O que deverá ser entregue:

1 **Código fonte bem indentado e comentado**: Código fonte contendo todas as especificações da etapa de Implementação e estruturas/funções bem definidas e tratadas.

2 **Documentação do Trabalho**: Relatório do trabalho em formato PDF seguindo o modelo fornecido (disponível no Moodle).

Comentários Gerais

- Toda e qualquer mensagem de orientação e de erro deve ser adequadamente tratada.
- Clareza, indentação e comentários no programa serão bem avaliados.
- O trabalho é para ser desenvolvido individualmente. Trabalhos identificados como cópias ou plágio terão a nota **dividida** ou **zerada** entre os alunos.
- Serão sorteados 3 alunos para apresentar e discutir seu trabalho com a turma.

Avaliação

1 Código bem indentado, comentado e funcional: 4 pts

2 Uso correto de variáveis, funções, estruturas de dados, arquivos e alocação dinãmica: 1 pt

3 Utilização de TADs e funções bem definidas sem repetição de código: 1 pt

4 Modularização: 0,5 pts

5 Código disponível em repositório do Github: 0,5 pts

6 Documentação utilizando modelo: 3 pts (Introdução 1 pt, Desenvolvimento 1 pt, Conclusão 0,5 pts, Referências e Apêndices 0,5 pts)