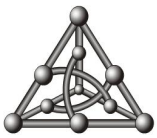


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

# Estruturas de Dados
## Trabalho 1 — Gerenciamento de Emergências

## 1 Descrição

Um hospital recebe chamadas de emergência e precisa priorizar o atendimento de pacientes de acordo com a gravidade do caso. Além disso, a equipe de atendimento do hospital deve ser capaz de buscar rapidamente o histórico de cada paciente para tomar decisões baseadas em registros médicos prévios.

Uma maneira de implementar um sistema para solucionar o problema de gerenciamento de emergências do hospital se baseia em duas principais ideias:

- Chamadas de emergência: uma lista de prioridades é usada para gerenciar as chamadas de emergência, onde cada chamada é classificada de acordo com o nível de gravidade. A gravidade é atribuída como um valor numérico: quanto maior o número, maior a gravidade. Assim, o sistema precisa sempre atender primeiro as chamadas mais críticas. Ou seja, as chamadas de emergência são armazenadas na lista de prioridades e a operação principal é a inserção de novas chamadas e a extração do paciente mais urgente.
- Registros médicos: uma tabela de dispersão é usada para armazenar os registros médicos dos pacientes, permitindo uma busca rápida pelo seu histórico, onde a chave é o número de identificação do paciente, mais outras informações como nome e enfermidade.

Para atendimento das emergências do hospital, o fluxo de trabalho do seu sistema deve seguir os seguintes passos:

1. Um paciente faz uma chamada de emergência.
2. A gravidade da emergência é avaliada e inserida na lista de prioridades.
3. O sistema prioriza o atendimento de acordo com a gravidade, sempre atendendo o caso mais urgente.

## 2 Programa, entrada e saída

Um número inteiro $n$ é fornecido inicialmente, indicando a quantidade de pacientes a serem cadastrados. A cada paciente são associadas as seguintes informações: identificação, nome e enfermidade. Esse cadastro é implementado como uma tabela de dispersão com **endereçamento aberto**. Use o valor devolvido por sua função de dispersão como identificação do paciente. Como sugestão, você pode usar a somar os valores dos caracteres na tabela ASCII do nome do paciente como o valor da chave correspondente. Considere o nome do paciente como uma sequência de até 50 caracteres, finalizada por uma vírgula.

Em seguida, as $n$ chamadas de emergência são fornecidas, sempre como um par de números inteiros onde o primeiro é a identificação do paciente e o segundo é a gravidade da enfermidade (um inteiro entre 0 e 10). Essa relação de chamadas é implementada como uma lista de prioridades, onde a prioridade é definida pela gravidade da enfermidade.

Por fim, é apresentada uma lista de atendimentos a serem realizados, respeitando as urgências.

## 2.1 Exemplo de entrada e saída

Um exemplo de cadastro de pacientes é mostrado a seguir.

```
16
Joao Carlos Silva, Infarto
Celia Maria Dotta Almeida, Gastrite
Davi Dantas da Costa, Dengue
Sergio Henrique Fernandes, Pneumonia
Jose Marques, Covid
Sara Veiga Oliveira, Pneumonia
Eliezer Santos da Silva, Encefalite
Maria Lima, Trombose
Ana Clara Santos, Sarampo
Miguel Loureiro, Asma
Artur Ferreira, Toxoplasmose
Helena Rodrigues, Leucemia,Alice Ferreira Pereira, Hepatite
Sandra Aparecida Marques, Covid
Guilherme Antonio de Oliveira, Enxaqueca
Sonia Souza, Osteoporose
```

Dado o cadastro acima, suponha que a lista de identificações atribuídas pela tabela de dispersão (chaves) e os nomes dos pacientes é dada da seguinte forma.

```
0 Guilherme Antonio de Oliveira
1 Jose Marques
2 Alice Ferreira Pereira
4 Sergio Henrique Fernandes
5 Miguel Loureiro
6 Sara Veiga Oliveira
7 Davi Dantas da Costa
8 Sonia Souza
9 Artur Ferreira
12 Ana Clara Santos
13 Maria Lima
14 Celia Maria Dotta Almeida
17 Sandra Aparecida Marques
18 Eliezer Santos da Silva
19 Joao Carlos Silva
22 Helena Rodrigues
```

Em seguida, uma relação de $n$ atendimentos, dados pela identificação do paciente e a urgência, é apresentada a seguir.

```
8 5
19 10
17 8
1 10
6 9
22 10
7 7
5 3
0 2
2 4
18 6
4 9
9 5
12 2
13 4
14 1
```

Por fim, a lista de atendimentos de pacientes, respeitando as urgências, é apresentada abaixo.

```
Joao Carlos Silva
Helena Rodrigues
Jose Marques
Sara Veiga Oliveira
Sergio Henrique Fernandes
Sandra Aparecida Marques
Davi Dantas da Costa
Eliezer Santos da Silva
Sonia Souza
Artur Ferreira
Alice Ferreira Pereira
Maria Lima
Miguel Loureiro
Guilherme Antonio de Oliveira
Ana Clara Santos
Celia Maria Dotta Almeida
```

## 3 Orientações gerais

O programa pode ser desenvolvido usando programação estruturada OU pode ser desenvolvido usando orientação a objetos. Caso opte por programação estruturada, use a linguagem C padrão. Caso escolha programação orientada a objetos, use a linguagem C++. Caso escolha a linguagem C++ você tem um bônus automático de `+0,5` ponto na sua Prova 1.

Não use códigos/funções/classes já existentes, especialmente caso escolha uma linguagem orientada a objetos. NÃO É PERMITIDO usar qualquer estrutura de dados já implementada, como por exemplo em bibliotecas do C++ ou de terceiros (**`vector`**, **`string`**, etc). Portanto, você deve implementar todas as estruturas de dados necessárias para seu programa.

Você DEVE, ao final do seu programa, liberar toda memória alocada dinamicamente tais como vetores, listas, e outras, e também garantir que seu programa não realiza acessos inválidos à memória. Para verificar essa questão, será utilizado o utilitário *Valgrind*, com o comando

```
valgrind --leak-check=full --show-reachable=yes --track-fds=yes ./meu_programa meu_codigo.cpp
```

(ou **`meu_codigo.c`**, se for o caso). Embora execute mais lentamente, você pode **compilar** seu programa com a opção de depuração **`-g`**, permitindo que o **`valgrind`** detalhe mais a saída, incluindo os números das linhas de seu programa com eventuais problemas.

O uso das estruturas de dados conforme as especificações propostas é OBRIGATÓRIO.

# 4 Entrega

Instruções para entrega do seu trabalho:

1. **Cabeçalho**

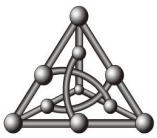
   Seu trabalho deve ter um cabeçalho com o seguinte formato:

   ```
   /**********************************************
   *
   * Nome do(a)(s) estudante(s)
   * Trabalho 1
   * Professor(a): Nome do(a) professor(a)
   *
   */
   ```

   ATENÇÃO! Se você fizer o trabalho 1 em dupla, o trabalho 2 deve ser feito com a mesma dupla, a não ser que um dos membros desista do curso. Neste último caso, o membro restante deve entregar o trabalho sozinho. NÃO troque de dupla durante o curso!

2. **Compilador**

   Para a correção do trabalho, será utilizado o compilador da linguagem C++ da coleção de compiladores GNU **`g++`**, com as opções de compilação **`-Wall -pedantic -std=c++11`** ao corrigir os programas. Ou o compilador da linguagem C da coleção de compiladores GNU **`gcc`**, com as opções de compilação **`-Wall -pedantic -ansi`**. Opcionalmente, você pode utilizar a flag **`-g`** para compilar seu programa e testá-lo pelo **`valgrind`**. Antes de entregar seu programa, verifique se ele tem extensão **`.cpp`** ou **`.c`**, compila sem mensagens de alerta e executa corretamente e não possui problemas de memória acusados pelo **`valgrind`**.





3. **Forma de entrega**

   A entrega será realizada diretamente na página da disciplina no AVA/UFMS. Um fórum de discussão deste trabalho já se encontra aberto. Após abrir uma sessão digitando seu *login* e sua senha, vá até o tópico "Trabalhos", e escolha "Trabalho 1" e "Entrega". Você pode entregar o trabalho quantas vezes quiser até às **23 horas e 59 minutos** do dia **3 de outubro de 2024**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos.

4. **Atrasos**

   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, falha de conexão com a internet, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

5. **Erros**

   Trabalhos com erros de compilação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação.

6. **O que entregar?**

   Você, ou sua dupla, deve entregar um único arquivo contendo APENAS o seu programa fonte com um nome significativo, como por exemplo, `controle_emergencias.cpp`. NÃO entregue qualquer outro arquivo, tal como o programa executável, já compilado.

7. **Verificação dos dados de entrada**

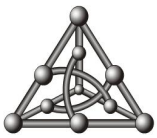
   Não se preocupe com a verificação dos dados de entrada do seu programa. Seu programa não precisa fazer consistência dos dados de entrada. Isto significa que se, por exemplo, o seu programa pede um número entre 1 e 10 e o usuário digita um número negativo, uma letra, um cifrão, etc, o seu programa pode fazer qualquer coisa, como travar o computador ou encerrar a sua execução abruptamente com respostas erradas.

8. **Arquivo com o programa-fonte**

   Seu arquivo contendo o programa-fonte deve estar bem organizado. Um programa tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso. Não esqueça da documentação[1] de seu programa e de suas funções/métodos.

   Dê um nome significativo para seu programa e adicione a extensão adequada (`.cpp` ou `.c`) a este arquivo. Por exemplo, `controle_emergencias.cpp` é um nome válido.

[1] Você sabe o que é documentação de um programa?





9. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE ou EM DUPLA. Cada estudante ou dupla tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir ideias sobre o trabalho, ao vivo ou no fórum de discussão da disciplina, mas NÃO copie o programa!

   Trabalhos considerados plagiados terão nota ZERO.