Pontifícia Universidade Católica de Campinas

Engenharia da Computação

Estrutura de Dados – Relatório 1

Membros do Grupo:

Enzo Eiki Yasuda - 24000880

Felipe Martins Leivas – 24001643

Gustavo Bordin Correa – 24002887

Campinas, 2025

Introdução

Neste projeto, foi nos proposto o desenvolvimento de um programa em linguagem C que representasse um sistema de uma clínica veterinária. O objetivo principal é desenvolver e aprofundar nossos conhecimentos e práticas com Filas, portanto, o sistema deverá ter manipulação de Filas.

Ademais, o sistema deverá trabalhar com 7 funções obrigatórias: inserir um PET na fila, atender um PET, buscar um PET, imprimir um relatório com todos os PETs a serem atendidos, imprimir informações do próximo PET a ser atendido, imprimir as informações de todos os PETs já atendidos e encerrar o sistema.Cada PET deverá receber informções sobre o mesmo, sendo elas: ID (pré-requisito ser um único), Nome, Espécie, Idade, Data de nascimento e Prioridade (Se o PET está na fila Emergencial ou Normal).

Apresentação do projeto

Começando o projeto, criamos o esqueleto do programa, isto é, três filas (normal, emergencial e atendidos), que derão base ào banco de dados dos PETs, e um switch case principal, que permitirá o usuário escolher qual função ele utilizará dentre as já mencionadas.

Sobre as filas:

A construção das filas foram feitas a partir das funções de struct guardadas no header: “Fila.h”, a qual tanto o arquivo main, quanto o próprio header podem chamar as diversas funções relacionadas a Fila e outras funções que seriam criadas posteriormente. Ao montar as filas, encontramos o primeiro impasse ao questionar sobre como guardar todas as informações do PET em um único espaço de memória alocada da fila. No entanto, logo entendemos que seria possível alocar uma estrutura inteira contendo todas as informações em um espaço na fila, assim resolvendo o dilema.

Sobre as estruturas:

Criando a estrutura “info_pet”, foram adicionadas as variáveis onde guradariam as informações do PET, tendo como inteiro (Idade, ID e Prioridade), string (nome e espécie) e uma estrutura, nomeada: “struct Data”, onde iria armazenar a data de nascimento do tipo inteiro (dia, mês e ano).

Após o desenvolvimento do esqueleto do programa, foi planejada toda parte funcional, a qual seria o conteúdo de cada case tendo todas as funções necessárias.

1- Inserir um PET:

A função começa com a escolha da prioridade do PET, depois todas as outras informções, as quias vão ser guardadas variáveis específicas para cada. Foi gerado o ID aleatoriamente, e as variáveis foram eviadas para a função “InsereFila(Fila, ...)”;, a qual copiará as informações recebidas na estrutura do nó, enviando assim para a fila escolhida no começo da função.

1.2 – ID:

Para inserir o ID, era necessário gerar um inteiro aleatório e garantir que este ID seria único, ou seja, não repetir em nenhuma fila. Para isso, criamos a função “int randomID (...)” que usa o relógio do sistema com o comando “srand(time(NULL));” e finalmente gera um número aleatório de 100 a 999 com a linha “int rd_num = 100 + (rand() % 900);”.

Em seguida, todas as filas são conferidas um a um com o ID gerado randomicamente. Caso seja repetido, usamos uma recursividade que chama a mesma função novamente criando outro ID. Assim que as filas forem completamente checadas de não ser repetido, retornamos a variável ID gerada.

2- Atender um PET:

Para atender um PET, foi preciso conferir se a fila emergencial estaria vazia, tendo ela como prioridade de atendimento. As informações do PET são retiradas usando a função “RetiraFila”, que remove o topo e retorna para o nó criado para receber as informações: “Imprimir”. Este guarda as informações e é enviado para a função “ImprimeFila” para imprimir as informações recebidas na tela do usuário, e o nó é inserido na fila de atendidos pela função “InsereFila”.

3- Buscar um PET:

Este foi o maior desafio do projeto, pois seria necessário comparar todas as filas, dentre nome ou ID, o que teria inúmeras verificações necessárias, dependendo do tamanho das filas. Todavia, foi possível reduzir ao mínimo as verificações.

Para realizar a função, foi perguntado se o cliente gostaria de procurar o PET pelo nome ou pelo ID. Caso for escolhida a opção “Nome”, é pedido o nome a ser procurado chamando a função “CorrigirString”, que em suma, transforma a posição 0 do “vetor de caracteres” em maiúscula (comando toupper), enquanto os demais caracteres são transformados em minúsculas (tolower), e caso tenha espaço ou traço, o próximo caracter será em maiúsculo. Assim, evitando qualquer tipo de problema no sistema.

Após isto, foi passado por todas as filas conferindo o ID ou comparando os nomes usando “strcmp”, dependendo da escolha do usuário. Toda vez que essa comparação retornar 1, ou seja, sempre que for igual, usamos a função “ImprimeFila”, até acabarem as opções (aux == NULL).

4- Imprimir Relatório de PETs a serem atendidos:

Esta foi uma tarefa bem mais simples, ainda mais em contraste com a última. Foi criada a função “ImprimeRelatorio” que usa continuamente a função” ImprimeFila” até chegar no fim da fila, imprimindo primeiro a fila emergencial e depois a normal.

5- Imprimir próximo PET a ser atendido:

Neste caso, achamos mais otimizado conferir se as filas estão vazias antes de imprimir qualquer coisa. Caso a emergencial estiver vazia, o sistema avisa que é um atendimento normal e imprime o primeiro espaço da fila normal, caso contrário, é impresso o primeiro espaço da fila emergencial.

Porém, se as duas filas estiverem vazias, o sistema avisa que ambas as filas estão vazias, ou seja, não existem PETs a serem atendidos.

6- Imprimir todos os PETs já atendidos:

Agora para imprimir todos os PETs atendidos, repetimos o processo da funcionalidade 4, mas usamos a função “ImprimeRelatório” apenas para a fila

Atendidos. Antes disso, conferimos se a fila de atendidos estaria vazia, caso seja verdade, é impresso um aviso dizendo que a fila está vazia.

7- Encerrar o programa:

Finalmente para encerrarmos o programa incrementamos a variável "encerrador" que sai do while, que estava englobando tanto o menu, quanto o switch case e, para poder então finalizar o sistema, liberamos todas as filas, para garantir que não fique armazenado informações desnecessárias na memória. Então imprimimos um rodapé de encerramento e o sistema retorna 0.

Considerações finais:

Durante toda a confecção do programa, nos asseguramos de deixar o sistema o mais "user friendly" possível, isto é, focamos em sempre diminuir ao máximo a chance do usuário, sem querer ou por falta de conhecimento, danificar o sistema ou encontrar algum erro.

Podemos ver essa adaptação quando corrigimos todas as strings, sempre que conferimos se a fila está vazia, usando a função "default" no switch case, quando corrigimos na função "Escolha3" a string digitada para informar se a busca será por nome ou ID, garantindo que não existam erros de digitação e que também pergunta repetidamente até que o usuário envie uma string compatível (Nome ou Id).

Referências Bibliográficas:

- Aulas da professora Lucia Filomena de Almeida Guimaraes.
- Slides de revisão em C da professora Lucia
- Blog da comunidade "Stack overflow" para entender sobre geração de números aleatórios através do relógio do sistema (link)