Trabalho Prático I - Estrutura de Dados II
Aluno: Mauricio Benjamin da Rocha
Dupla: Lazaro Claubert de Sousa Rodrigues Oliveira

Resumo
O presente trabalho tem como objetivo desenvolver um sistema de cadastro de séries e suas características específicas como temporadas, e participantes envolvidos nas mesmas. O sistema foi desenvolvido em linguagem C usando conceitos básicos e avançados visando garantir a qualidade do sistema.

Introdução

As séries são compostas por seu código de identificação, título, número de temporadas e suas respectivas temporadas. A organização das temporadas foi feita em formato tanto de árvore binária de busca quanto de árvore AVL visando analisar qual a mais adequada para solucionar o problema em meio a um maior âmbito de possibilidades.

As temporadas são constituídas por seu número da temporada, título, quantidade de episódios, ano de lançamento e uma lista contendo todos os participantes envolvidos para a realização da mesma. Conforme foi dito anteriormente a organização dos participantes é feita por uma lista simples permitindo acesso as todas as informações fornecidas sobre os participantes.

As informações fornecidas pelos participantes são: nome do artista, nome do personagem e uma breve descrição do personagem atuado.

O sistema foi desenvolvido em linguagem C usando conceitos básicos e avançados visando garantir a qualidade do sistema. O sistema armazena as séries e temporadas em árvores binárias, já os participantes são salvos em uma lista simples.

Seções Específicas

Esta seção tem como objetivo explicar como funcionam as estruturas de dados usadas e as funções do sistema.

Estruturas de Dados das Árvores Binárias de Busca

O sistema utiliza três estruturas de dados principais:

Serie: Representa uma série de TV e contém informações como código, título e o número de temporadas. Cada série também possui uma árvore binária de temporadas. As séries são organizadas de forma a montarem uma árvore binária ao final;

Temporada: Representa uma temporada de uma série e contém informações como número da temporada, título, quantidade de episódios, ano e uma lista de participantes (atores).

Participante: Representa um participante, que pode ser um ator, e inclui detalhes como nome do artista, nome do personagem e descrição do personagem.

Funções Usando Árvore Binária de Busca:

- int iniciarS(Serie **s)
  Propósito: Inicializa uma estrutura de dados para a série.
  Características:
    - Recebe a referência de um ponteiro série **s.
    - Aloca memória para a série.
    - Retorna um valor inteiro indicando sucesso ou falha na inicialização.
- void preencherS(Serie **s, int id, char titulo[])
  Propósito: Preenche os detalhes de uma série, incluindo seu código e título.
  Características:
    - Recebe a referência de um ponteiro série **s, um ID e um título.
    - Aloca memória para uma nova série.
    - Preenche o código e o título da série.

Define o número de temporadas e inicia a lista de temporadas como nula.

- void mostrarS(Serie *s)
  Propósito: Exibe informações de uma série.
  Características:
  Recebe um ponteiro para uma série *s.
  Imprime o código, título e número de temporadas da série.

- int cadastrarS(Serie **s, Serie *novo)
  Propósito: Cadastra uma série na estrutura de dados de séries, organizando-as em uma árvore.
  Características:
  Recebe a referência de um ponteiro série **s e uma série *novo.
  Organiza as séries com maior código para a direita e menores para a esquerda em uma árvore.
  Retorna um valor inteiro indicando sucesso ou falha no cadastro.

- int validarS(Serie *s, int id)
  Propósito: Valida a existência de uma série com um código específico.
  Características:
  Recebe um ponteiro para uma série *s e um ID.
  Verifica se uma série com o ID especificado existe na estrutura de dados.
  Retorna um valor inteiro indicando a validade.

- Serie *buscarS(Serie *s, int id)
  Propósito: Busca uma série com um código específico e a retorna.
  Características:
  Recebe um ponteiro para uma série *s e um ID.
  Realiza uma busca na estrutura de dados de séries.
  Retorna um ponteiro para a série encontrada ou nulo se não for encontrada.

- int geraId(Serie **s)
  Propósito: Gera um ID aleatório para uma série que não existe.
  Características:
  Recebe a referência de um ponteiro série **s.
  Gera um ID aleatório e verifica se ele já existe em alguma série.
  Retorna um ID único.

- void mostrar_all_S(Serie **s)
  Propósito: Exibe todas as séries cadastradas.
  Características:
  Recebe a referência de um ponteiro série **s.
  Realiza uma travessia em ordem na árvore de séries e imprime os detalhes de cada série.

- void liberar_all_P(Participante **p)
  Propósito: Libera a memória alocada para a lista de participantes de uma temporada.
  Características:
  Recebe a referência de um ponteiro participante **p.
  Realiza a liberação recursiva da memória alocada para a lista de participantes.

- void liberar_all_T(Temporada **t)
  Propósito: Libera a memória alocada para as temporadas de uma série.
  Características:
    Recebe a referência de um ponteiro temporada **t.
    Realiza a liberação recursiva da memória alocada para as temporadas, incluindo a lista de participantes.

- void liberar_all_S(Serie **s)
  Propósito: Libera a memória alocada para todas as séries.
  Características:
    Recebe a referência de um ponteiro série **s
    Realiza a liberação recursiva da memória alocada para todas as séries, incluindo temporadas e participantes.
    Dispara as funções liberar_all_T e liberar_all_P

- void preencherT(Temporada **t, int id)
  Propósito: Preenche os detalhes de uma temporada, incluindo título, número de episódios, ano e atores.
  Características:
    Recebe a referência de um ponteiro temporada **t e um ID.
    Aloca memória para uma nova temporada.
    Preenche as informações da temporada, incluindo a lista de participantes (atores).

- int cadastrarT(Temporada **t, int id)
  Propósito: Cadastra uma temporada em uma série, organizando-as em uma árvore.
  Características:
    Recebe a referência de um ponteiro temporada **t e um ID.
    Organiza as temporadas com maior número para a direita e menores para a esquerda em uma árvore.
    Retorna um valor inteiro indicando sucesso ou falha no cadastro.

- void mostrarT(Temporada *t)
  Propósito: Exibe informações de uma temporada.
  Características:
    Recebe um ponteiro para uma temporada *t.
    Imprime o número da temporada, título, quantidade de episódios e ano.

- Temporada *buscarT(Temporada *t, int id)
  Propósito: Busca uma temporada com um número específico e a retorna.
  Características:
    Recebe um ponteiro para uma temporada *t e um ID.
    Realiza uma busca na estrutura de dados de temporadas.
    Retorna um ponteiro para a temporada encontrada ou nulo se não for encontrada.

- void mostrar_all_T(Temporada *t)
  Propósito: Exibe todas as temporadas de uma série.
  Características:
    Recebe um ponteiro para uma temporada *t.

Realiza uma travessia em ordem na árvore de temporadas e imprime os detalhes de cada temporada.

- void mostrarP(Participante *p)
  Propósito: Exibe informações de um participante.
  Características:
  Recebe um ponteiro para um participante *p.
  Imprime o nome do artista, nome do personagem e descrição do personagem.

- void mostrar_all_P(Participante *p)

  Propósito: Exibe todos os participantes de uma temporada.
  Características:
  Recebe um ponteiro para uma lista de participantes *p.
  Realiza uma travessia em ordem na lista de participantes e imprime os detalhes de cada participante.

- void letra_D(Temporada *t)
  Propósito: Mostrar todos os participantes de todas as temporadas de uma determinada série.
  Características:
  Recebe um ponteiro para uma temporada *t.
  Realiza uma travessia em pós ordem na árvore de temporadas e para cada temporada, direciona a mesma para a função mostrar_all_P para que possa mostrá-los.

Estruturas de Dados das Árvores AVL

O sistema utiliza três estruturas de dados principais:

Serie: Representa uma série de TV e contém informações como código, título e o número de temporadas. Cada série também possui uma árvore binária de temporadas. As séries são organizadas de forma a montarem uma árvore binária ao final;

Temporada: Representa uma temporada de uma série e contém informações como número da temporada, título, quantidade de episódios, ano e uma lista de participantes (atores).

Participante: Representa um participante, que pode ser um ator, e inclui detalhes como nome do artista, nome do personagem e descrição do personagem.

Funções Usando Árvore Binária AVL:

- int altura_Serie(Serie *s)
  Propósito: Obter a altura de um nó na árvore de séries.
  Características:
  Recebe um ponteiro para uma série *s.
  Retorna a altura do nó passado.

- int calcularFB_Serie(Serie *s)
  Propósito: Calcular o fator de balanceamento de um nó na árvore de séries.
  Características:
  Recebe um ponteiro para uma série *s.
  Retorna o fator de balanceamento do nó.

- void atualizarAltura_Serie(Serie *s)
  Propósito: Atualizar a altura de um nó na árvore de séries.

Características:
Recebe um ponteiro para uma série *s.
Atualiza o campo altura do mesmo.

- Serie *rotacaoDireita_Serie(Serie *s)
  Propósito: Realizar a rotação para a direita do nó na árvore de séries.
  Características:
  Recebe um ponteiro para uma série *s.
  Retorna o novo nó “raiz” após a rotação para a direita do nó passado.

- Serie *rotacaoEsquerda_Serie(Serie *s)
  Propósito: Realizar a rotação para a direita do nó na árvore de séries.
  Características:
  Recebe um ponteiro para uma série *s.
  Retorna o novo nó “raiz” após a rotação para a esquerda do nó passado.

- int iniciarS(Serie **s)
  Propósito: Inicializa uma estrutura de dados para a série.
  Características:
  Recebe a referência de um ponteiro série **s.
  Aloca memória para a série.
  Retorna um valor inteiro indicando sucesso ou falha na inicialização.

- void preencherS(Serie **s, int id, char titulo[])
  Propósito: Preenche os detalhes de uma série, incluindo seu código e título.
  Características:
  Recebe a referência de um ponteiro série **s, um ID e um título.
  Aloca memória para uma nova série.
  Preenche o código e o título da série.
  Define o número de temporadas e inicia a lista de temporadas como nula.

- void mostrarS(Serie *s)
  Propósito: Exibe informações de uma série.
  Características:
  Recebe um ponteiro para uma série *s.
  Imprime o código, título e número de temporadas da série.

- int cadastrarS(Serie **s, Serie *novo)
  Propósito: Cadastra uma série na estrutura de dados de séries, organizando-as em uma árvore.
  Características:
  Recebe a referência de um ponteiro série **s e uma série *novo.
  Organiza as séries com maior código para a direita e menores para a esquerda em uma árvore.
  Retorna um valor inteiro indicando sucesso ou falha no cadastro.

- int validarS(Serie *s, int id)
  Propósito: Valida a existência de uma série com um código específico.
  Características:
  Recebe um ponteiro para uma série *s e um ID.

Verifica se uma série com o ID especificado existe na estrutura de dados.
Retorna um valor inteiro indicando a validade.

- Serie *buscarS(Serie *s, int id)
  Propósito: Busca uma série com um código específico e a retorna.
  Características:
  Recebe um ponteiro para uma série *s e um ID.
  Realiza uma busca na estrutura de dados de séries.
  Retorna um ponteiro para a série encontrada ou nulo se não for encontrada.

- int geraId(Serie **s)
  Propósito: Gera um ID aleatório para uma série que não existe.
  Características:
  Recebe a referência de um ponteiro série **s.
  Gera um ID aleatório e verifica se ele já existe em alguma série.
  Retorna um ID único.

- void mostrar_all_S(Serie **s)
  Propósito: Exibe todas as séries cadastradas.
  Características:
  Recebe a referência de um ponteiro série **s.
  Realiza uma travessia em ordem na árvore de séries e imprime os detalhes de cada série.

- void liberar_all_P(Participante **p)
  Propósito: Libera a memória alocada para a lista de participantes de uma temporada.
  Características:
  Recebe a referência de um ponteiro participante **p.
  Realiza a liberação recursiva da memória alocada para a lista de participantes.

- void liberar_all_T(Temporada **t)
  Propósito: Libera a memória alocada para as temporadas de uma série.
  Características:
  Recebe a referência de um ponteiro temporada **t.
  Realiza a liberação recursiva da memória alocada para as temporadas, incluindo a lista de participantes.

- void liberar_all_S(Serie **s)
  Propósito: Libera a memória alocada para todas as séries.
  Características:
  Recebe a referência de um ponteiro série **s
  Realiza a liberação recursiva da memória alocada para todas as séries, incluindo temporadas e participantes.
  Dispara as funções liberar_all_T e liberar_all_P

- int altura_Temporada(Temporada *t)
  Propósito: Obter a altura de um nó na árvore de temporadas.
  Características:
  Recebe um ponteiro para uma temporada *t.
  Retorna a altura do nó passado.

- int calcularFB_Temporada(Temporada *t)
    Propósito: Calcular o fator de balanceamento de um nó na árvore de temporadas.
    Características:
        Recebe um ponteiro para uma temporada *s.
        Retorna o fator de balanceamento do nó.

- void atualizarAltura_Temporada(Temporada *t)
    Propósito: Atualizar a altura de um nó na árvore de temporadas.
    Características:
        Recebe um ponteiro para uma temporada *t.
        Atualiza o campo altura do mesmo.

- Temporada *rotacaoDireita_Temporada(Temporada *t)
    Propósito: Realizar a rotação para a direita do nó na árvore de temporadas.
    Características:
        Recebe um ponteiro para uma temporada *t.
        Retorna o novo nó “raiz” após a rotação para a direita do nó passado.
- Temporada *rotacaoEsquerda_Temporada(Temporada **t)
    Propósito: Realizar a rotação para a direita do nó na árvore de temporadas.
    Características:
        Recebe um ponteiro para uma temporada *t.
        Retorna o novo nó “raiz” após a rotação para a esquerda do nó passado.

- void troca(Participante **a, Participante **b)
    Propósito: Realizar a troca de posição do participante ‘a’ pelo participante ‘b’.
    Características:
        Recebe a referência de 2 ponteiros, um para ‘a’ e outro para ‘b’.
        ‘a’ se torna ‘b’ e ‘b’ se torna o ‘a’ original.

- void OrdenaAtores(Participante **head)]
    Propósito: Ordenar os atores em ordem alfabética.
    Características:
        Recebe a referência de um ponteiro para a lista de participantes.
        Ordena a lista de participantes.

- void cadastran_atores(Participante **l, int n_atores)
    Propósito: Cadastrar todos os atores de uma temporada.
    Características:
        Recebe a referência de um ponteiro para a lista de participantes **l.
        Repete o cadastro até que número de atores chegue a zero.

- void preencherT(Temporada **t, int id)
    Propósito: Preenche os detalhes de uma temporada, incluindo título, número de episódios, ano e atores.
    Características:
        Recebe a referência de um ponteiro temporada **t e um ID.
        Aloca memória para uma nova temporada.

Preenche as informações da temporada, incluindo a lista de participantes (atores).

- int cadastrarT(Temporada **t, int id)
  Propósito: Cadastra uma temporada em uma série, organizando-as em uma árvore.
  Características:
  Recebe a referência de um ponteiro temporada **t e um ID.
  Organiza as temporadas com maior número para a direita e menores para a esquerda em uma árvore.
  Retorna um valor inteiro indicando sucesso ou falha no cadastro.

- void mostrarT(Temporada *t)
  Propósito: Exibe informações de uma temporada.
  Características:
  Recebe um ponteiro para uma temporada *t.
  Imprime o número da temporada, título, quantidade de episódios e ano.

- Temporada *buscarT(Temporada *t, int id)
  Propósito: Busca uma temporada com um número específico e a retorna.
  Características:
  Recebe um ponteiro para uma temporada *t e um ID.
  Realiza uma busca na estrutura de dados de temporadas.
  Retorna um ponteiro para a temporada encontrada ou nulo se não for encontrada.

- void mostrar_all_T(Temporada *t)
  Propósito: Exibe todas as temporadas de uma série.
  Características:
  Recebe um ponteiro para uma temporada *t.
  Realiza uma travessia em ordem na árvore de temporadas e imprime os detalhes de cada temporada.

- void mostrarP(Participante *p)
  Propósito: Exibe informações de um participante.
  Características:
  Recebe um ponteiro para um participante *p.
  Imprime o nome do artista, nome do personagem e descrição do personagem.

- void mostrar_all_P(Participante *p)
  Propósito: Exibe todos os participantes de uma temporada.
  Características:
  Recebe um ponteiro para uma lista de participantes *p.
  Realiza uma travessia em ordem na lista de participantes e imprime os detalhes de cada participante.

- void letra_D(Temporada *t)
  Propósito: Mostrar todos os participantes de todas as temporadas de uma determinada série.
  Características:

Recebe um ponteiro para uma temporada *t.
Realiza uma travessia em pós ordem na árvore de temporadas e para cada temporada, direciona a mesma para a função mostrar_all_P para que possa mostrá-los.

Resultados da Execução do Programa

Visando avaliar o desempenho dos algoritmos propostos foram executados testes de desempenho na máquina com a configuração conforme a Tabela 1, onde foram testadas as seguintes quantidades de dados: 1,10, 100, 1000 e 10000. As tabelas 2 e 3 apresenta a aproximação dos valores encontrados durante os testes usando a árvore binária de busca, tais valores possuem somente duas casas decimais e podem variar devido a fatores externos e/ou a ordem que elementos são adicionados na árvore. As tabelas 4 e 5 ilustram os resultados encontrados usando a árvore AVL.

Tabela 1: Hardware Usado

Processador
AMD Ryzen™ 5 3350G with Radeon™ Vega Graphics × 8

Tabela 2: Resultados obtidos aproximadamente (Binária de Busca)

Quantidade de elementos	Tempo médio de Inserção (ms)	Tempo total de Inserção (ms)
1	0,01	0,01
10	0,01	0,09
100	0,01	1,38
1000	0.07	73,00
10000	1.34	13404.36

Tabela 3: Resultados obtidos aproximadamente (Binária de Busca)

Quantidade de elementos	Tempo médio de Busca(ms)	Tempo total de Busca (ms)
1	0,09	0,09
10	0,03	0,26
100	0,03	2,5
1000	0.01	11,00
10000	0.01	123,00

Tabela 4: Resultados obtidos aproximadamente (AVL)

Quantidade de elementos	Tempo médio de Inserção (ms)	Tempo total de Inserção (ms)

1	0,05	0,05
10	0,01	0.12
100	0,01	1,02
1000	0.06	55.45
10000	1.17	11748.64

Tabela 5: Resultados obtidos aproximadamente (AVL)

Quantidade de elementos	Tempo médio de Busca(ms)	Tempo total de Busca (ms)
1	0,05	0,05
10	0,35	0,03
100	2,78	0,03
1000	0.01	11,28
10000	0.02	152,69

Conclusão

O sistema de gerenciamento de séries de TV apresenta um exemplo prático de como usar estruturas de dados e árvores binárias para organizar informações de forma eficiente. Ele permite ao usuário cadastrar séries, temporadas e atores, bem como visualizar e organizar essas informações. O experimento permitiu analisar a diferença de desempenho de ambas as estruturas de dados em diferentes cenários. Onde dependendo da quantidade de dados e operação realizada, determinada árvore possui desempenho melhor do que sua rival.

Apêndice

Juntamente com este relatório segue em anexo um zip contendo todos os códigos desenvolvidos nesse projeto. Dentro do arquivo zip existem 4 arquivos intitulados teste1, teste2, teste3 e teste4, onde os mesmos foram usados para executar os testes. Tal medida foi adotada visando uma melhor organização do relatório e da exposição do trabalho realizado.