UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

USP – ICMC – SSC

SSC0603 – Estrutura de Dados I (ED1) – 2025/2

TRABALHO TP03 Final – AVL index + LDE data

TP03 - AVL (Árvore Binária Ordenada e Balanceada) + LDE (Lista Dinâmica Encadeada)
TP03 - Trabalho INDIVIDUAL

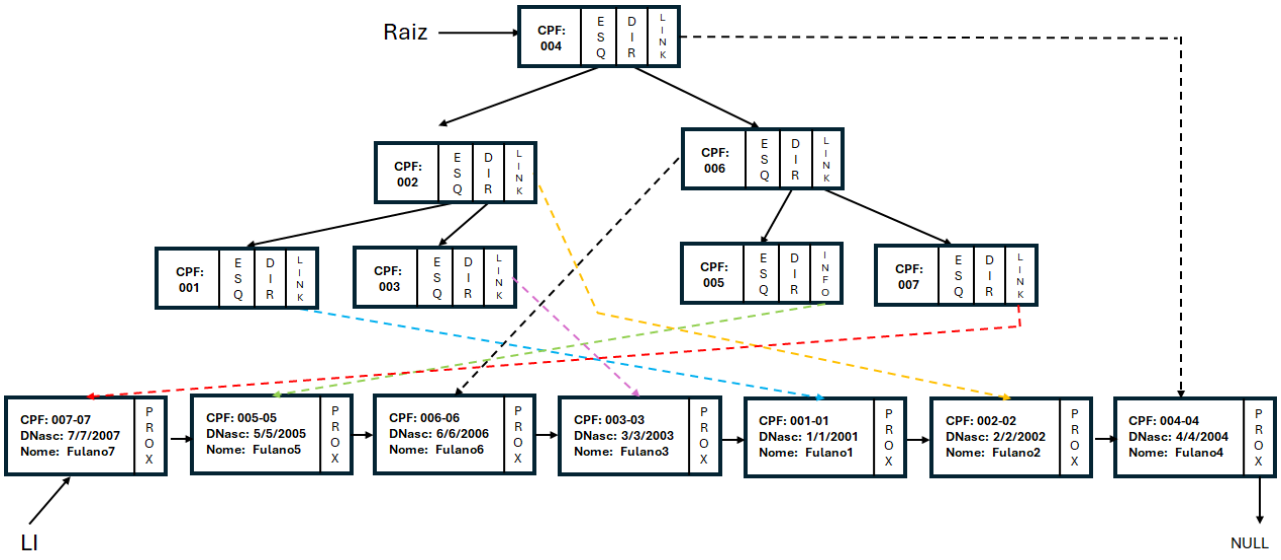
Versão 1.0r00 - VER sempre o PDF com descrição completa!
ATENÇÂO: DATA LIMITE DE UPLOAD 12/12/2025 (23h55) – Data setada no RunCodes

> Use listas **LDE Lista Encadeada Simples ou Dupla** neste trabalho [TAD Backes, FOsorio, ou outro]
> Use listas **AVL Árvore Binária Ordenada Balanceada** no trabalho [TAD Backes, FOsorio,ou outro]
>> Usar uma **AVL balanceada no TP03**, no TP03 será usada ArvBin Balanceada e Ordenada(AVL)

Este trabalho é uma "continuação do trabalho" TP02, porém substituindo o uso de ABO (não balanceada) por uma AVL (balanceada).

> Conceito do Programa proposto: Igual ao TP02, mas com AVL Balanceada
i) Criar uma árvore balanceada (AVL) de CPFs que vai servir de "índice" para o acesso rápido aos demais dados que estão em uma lista dinâmica encadeada LDE (lista linear) do tipo Simples ou Dupla.
ii) A ideia principal é que os nodos da AVL contenham apenas o CPF (9 dígitos, sem os 2 dígitos finais de verificação - DV), porém os nodos desta AVL irão conter também um "link" de acesso rápido aos dados inseridos em uma LDE (lista linear). Os dados complementares da pessoa irão ficar armazenados no nodo da LDE: CPF (9 dígitos), DV (2 dígitos), Data de Nascimento (dia, mês e ano) e Nome (max. 50 caracteres).
iii) Com isso, para encontrar um dado na AVL o tempo de acesso será bastante rápido (busca binária), e depois chegamos ao nodo da LDE em 1 passo apenas ("link" de acesso). Além disso, a inserção na LDE deve ser feita no início, portanto, também precisa de apenas 1 passo para inserir os dados.
> Custo de inserção de 'n' nodos na AVL: O(N.LogN). Custo pior caso de inserção na LDE: O(1).
> Custo de consulta de CPF na AVL: O(LogN). Custo para acesso aos demais dados LDE: O(1).
>> Se a LDE fosse uma base de dados em disco, poderia usar as funções fseek/ftell para acesso direto. 😊

O diagrama abaixo descreve como será a Estrutura de Dados que deve ser implementada:



A especificação é a mesma do TP02 em termos de estrutura de dados, apenas substituindo a ABO por AVL. Algumas saídas do programa são diferentes, com informações adicionais.

Qual o segredo desta estrutura?

- Ao inserir o nodo na lista encadeada (LDE), obter o ponteiro do nodo inserido
  E inserir este ponteiro no nodo da árvore binária ordenada e balanceada tipo AVL !
  (retorna o ponteiro do nodo LDE para inserção na AVL)

As duas estruturas de dados estarão conectadas, porém podemos fazer uma busca binária na árvore (percorrer a árvore), e depois acessar os dados do nodo mais "completo" na lista encadeada. A árvore sendo balanceada, permite uma inserção e consulta muito rápida, mesmo no pior caso!

Esta especificação deve ser respeitada para que o programa possa depois ser "reaproveitado", como por exemplo (como indicado acima), no acesso as informações com um "link" apontando para blocos em disco.

Quais dados o programa irá ler do disco?
Arquivo "basedados.txt" **ou** "basedados.csv" (descrição em anexo)
Será melhor avaliado o programa que usar o formato CSV. Uso de arquivo .TXT => desconto -0.3 pts.

O que deve ser gerado pelo programa?
Saida na tela: [prints]
<nro. de nodos da ABO>
<altura da árvore ABO>
<dados do 1º. nodo considerando a ordenação – 1º. na ordem>
<dados do último nodo considerando a ordenação – último na ordem>

Exemplo considerando o diagrama apresentado:
10
4
000000001-01 01/01/2001 Fulano1
000000010-10 10/10/2010 Fulan10

- Importante:
  Note que o CPF tem sempre 9 casas (com zeros a esquerda se necessário),
  Note que o DV tem sempre 2 casas (com zeros a esquerda se necessário),
  Note que a data é no formato DD/MM/AAAA (com zeros a esquerda se necessário).

O programa deve gravar os seguintes arquivos em disco:
dados1.txt => Gravar o conteúdo da lista encadeada LDE onde os dados são inseridos sempre no início
Descrição detalhada e exemplo do arquivo dados1.txt em anexo

dados2.txt => Gravar o conteúdo da árvore binária ordenada em modo "Em Ordem" (só CPF)
Descrição detalhada e exemplo do arquivo dados2.txt em anexo

dados3.txt => Gravar o conteúdo da árvore binária ordenada em modo "Pré-Ordem" (só CPF)
Os dados exibidos são o nodo e seus 2 filhos (esq e dir) COMO DESCRITO no ANEXO!
Descrição detalhada e exemplo do arquivo dados3.txt em anexo (SIGA O FORMATO DADO!)

O arquivo de entrada de dados "basedados.txt" possui os dados terminados pelo valor: -1

O usuário irá digitar (scanf) a opção do modo de funcionamento do programa:

Valores que podem ser digitados pelo usuário: 1 ou 2 (2 opções, 2 modos de funcionamento).
Estes dados são entrados (digitados) pelo teclado (como é usual no CodeBlocks) ou vem definidos no caso de teste (no RunCodes) como sendo a entrada do programa.

O usuário irá digitar (scanf) a opção do modo de funcionamento do programa:

1 <enter> => Executar o programa conforme descrito acima.
2 <enter> => Executar ao final do programa 3 linhas de código adicionais, logo antes de terminar.
system("cat dados1.txt");
system("cat dados2.txt");
system("cat dados3.txt");

Esta opção exibe os 3 arquivos de saída gerados!

ANEXO 1 – Exemplo do Arquivo de Entrada XLS (no TXT, substituir vírgula por enter)
CPF, DV, Dia,Mês,Ano,Nome

000000004,04,4,4,2004,Fulano4,
000000002,02,2,2,2002,Fulano2,
000000001,01,1,1,2001,Fulano1,
000000003,03,3,3,2003,Fulano3,
000000006,06,6,6,2006,Fulano6,
000000005,05,5,5,2005,Fulano5,
000000007,07,7,7,2007,Fulano7,
000000008,08,8,8,2008,Fulano8,
000000009,09,9,9,2009,Fulano9,
000000010,10,10,10,2010,Fulano10,
-1 FIM

ANEXO 2 - Formato do Arquivo de Dados1.txt
CPF-DV DD/MM/AAAA Nome (preencher com zeros CPF, DV, DD, MM

000000007-07 07/07/2007 Fulano7
000000005-05 05/05/2005 Fulano5
000000006-06 06/06/2006 Fulano6
000000003-03 03/03/2003 Fulano3
000000001-01 01/01/2001 Fulano1
000000002-02 02/02/2002 Fulano2
000000004-04 04/04/2004 Fulano4
000000008-08 08/08/2008 Fulano8
000000009-09 09/09/2009 Fulano9
000000010-10 10/10/2010 Fulano10

ANEXO 3 – Formato do Arquivo de Dados2.txt (Só CPF em ordem, com zeros na frente)

000000001
000000002
000000003
000000004
000000005
000000006
000000007
000000008
000000009
000000010

ANEXO 4 – Formato do Arquivo de Dados3.txt (Em pré-ordem, CPF E:FilhoEsq D:FilhoDir)
Formato: CPF E:CPF D:CPF (preencher com zeros, se não tem E: ou D: print NULL)

000000004 E: 000000002 D: 000000008
000000002 E: 000000001 D: 000000003
000000001 E: NULL D: NULL
000000003 E: NULL D: NULL
000000008 E: 000000006 D: 000000009
000000006 E: 000000005 D: 000000007
000000005 E: NULL D: NULL
000000007 E: NULL D: NULL
000000009 E: NULL D: 000000010
000000010 E: NULL D: NULL

(*) Obs.: conforme a implementação do algoritmo AVL, podem haver pequenas diferenças na forma como a árvore é construída e assim, resultar em uma saída que não seja exatamente igual a saída indicada acima (árvore balanceada, porém com estrutura e balanceamento diferente).

========
F.Osório
Nov.2025
========