

Atividade Prática 04

Manipulação de AVLs

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana
Curso de Engenharia de Computação
Disciplina de Estrutura de Dados 1 - EDCO3A
Prof. Dr. Rafael Gomes Mantovani

Instruções:

- Leia todas as instruções corretamente para poder desenvolver sua atividade/programa;
- Evite plágio (será verificado por meio de ferramentas automatizadas). Faça seu programa com os seus nomes de variáveis e lógica de solução. Plágios identificados anularão as atividades entregues de todos os envolvidos.
- Adicione comentários nos códigos explicando seu raciocínio e sua tomada de decisão. Porém, não exagere nos comentários, pois a própria estrutura do programa deve ser auto-explicativa.
- Salve sua atividade em um arquivo único, com todas as funções e procedimentos desenvolvidos. É esse **arquivo único** que deverá ser enviado ao professor.

1 Descrição da atividade

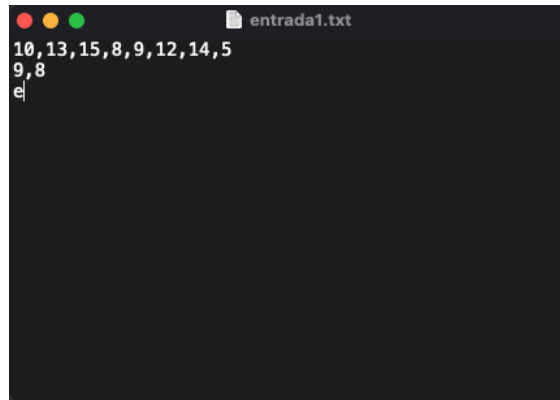
Explore e extrapole as implementações de árvores binárias balanceadas desenvolvidas em sala para implementar um programa que insira e remova chaves inteiras de uma árvore AVL.

2 Entradas do programa

O programa receberá dois arquivos texto como parâmetros de entrada:

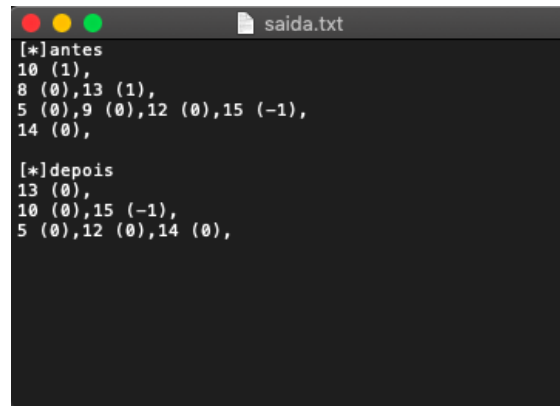
- **arquivo de entrada:** um arquivo texto simples com três linhas (Figura 1a). A primeira linha do arquivo apresenta em ordem as chaves que precisam ser inseridas na AVL. A segunda linha apresenta as chaves que serão removidas após todas as inserções. Por fim, a terceira linha contém um único caractere que define qual estratégia será usada para realizar as remoções: 'e' indica que a substituição será feita usando o **maior** elemento da sub-árvore esquerda, e 'd' indica o **menor** elemento da sub-árvore direita.

- **arquivo de saída:** é o arquivo onde serão impressos os estados de memória da AVL. Neste arquivo vocês devem imprimir duas árvores - a configuração da árvore após todas as inserções (antes); e a configuração da árvore após realizadas as remoções (depois). Atentem-se que a impressão no arquivo reflete um percurso em nível, um nível por linha, e indicando o grau de desbalanceamento de cada nó. A Figura 1b mostra as correspondentes árvores geradas pela entrada fornecida na Figura 1a.



```
10,13,15,8,9,12,14,5
9,8
e
```

(a) Exemplo de arquivo de entrada.



```
[*]antes
10 (1),
8 (0),13 (1),
5 (0),9 (0),12 (0),15 (-1),
14 (0),

[*]depois
13 (0),
10 (0),15 (-1),
5 (0),12 (0),14 (0),
```

(b) Exemplo de arquivo de saída.

Figura 1: Valores de entrada e correspondente arquivo de saída gerado pelo programa.

Para rodar o programa por linha de comando, manipular os argumentos **argc** e **argv** da função **main**. Para executar o programa por linha de comando, deve-se obedecer o seguinte padrão:

[nome do programa] [arquivo de entrada] [arquivo de saída]

Exemplo de execução de um programa chamado **avls.c**:

```
./avls entrada1.txt saida1.txt
```

3 Orientações gerais

Além da funcionalidade desejada, implementar também o controle de erros, para lidar com exceções que possam ocorrer, como por exemplo:

- problemas nas aberturas dos arquivos de entrada e saída;
- arquivo de entrada vazio (sem informação);
- arquivo de entrada fora do padrão esperado (opções inválidas para tipo da pilha, ou números que não sejam inteiros nas demais linhas);
- etc.

Opcionalmente, para acompanhamento do desenvolvimento, pode-se criar um repositório individual no `github`.

3.1 Critério de correção

A nota na atividade será contabilizada levando-se em consideração alguns critérios:

1. pontualidade na entrega;
2. não existir plágio;
3. completude da implementação (tudo foi feito);
4. o código compila e executa;
5. uso de `argc` e `argv` para controle dos arquivos de teste;
6. implementar o parser para entrada dos dados via arquivo texto;
7. implementação das duas estruturas necessárias;
8. legibilidade do código (identação, comentários nos blocos mais críticos);
9. implementação dos controles de erros (arquivos de entrada inválidos, e erros no programa principal);
10. controle de memória: chamar o destrutor e desalocar a memória de tudo se usar a estrutura dinâmica, fechar os arquivos, etc;
11. executar corretamente os casos de teste.

Em cada um desses critérios, haverá uma nota intermediária valorada por meio de conceitos: **Sim** - se a implementação entregue cumprir o que se esperava daquele critério; **Parcial** - se satisfizer parcialmente o tópico; e **Não** se o critério não foi atendido.

3.2 Dados para envio da atividade

Ao elaborar seu programa, crie um único arquivo fonte (.c) seguindo o padrão de nome especificado:

`ED1-<ANO>-<SEMESTRE>-AT04-AVLs-<NOME>.c`

Exemplo:

`ED1-2021-2-AT04-AVLs-RafaelMantovani.c`

A entrega da atividade será via Moodle: o link será disponibilizado na página da disciplina.

4 Links úteis

Arquivos em C:

- <https://www.inf.pucrs.br/~pinho/LaproI/Arquivos/Arquivos.htm>
- <https://www.geeksforgeeks.org/basics-file-handling-c/>
- <https://www.programiz.com/c-programming/c-file-input-output>

Argumentos de Linha de comando (argc e argv):

- https://www.tutorialspoint.com/cprogramming/c_command_line_arguments.htm
- <http://linguagemc.com.br/argumentos-em-linha-de-comando/>
- http://www.univasf.edu.br/~marcelo.linder/arquivos_pc/aulas/aula19.pdf
- http://www.inf.ufpr.br/cursos/ci067/Docs/NotasAula/notas-31_Argumentos_linha_comando.html
- <http://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node145.html>

Árvores AVL:

- https://en.wikipedia.org/wiki/AVL_tree
- <https://www.cs.usfca.edu/~galles/visualization/AVLtree.html>
- http://www.ufjf.br/jairo_souza/files/2009/12/5-Indexa~ao-Arvore-AVL.pdf
- <http://dcm.ffclrp.usp.br/~augusto/teaching/aedi/AED-I-Arvores-AVL.pdf>

Referências

- [1] Thomas H. Cormen,; Ronald Rivest; Charles E. Leiserson; Clifford Stein. Algoritmos - Teoria e Prática - 3^a Ed. Elsevier - Campus, 2012.
- [2] Nivio Ziviani. Projeto de algoritmos com implementações: em Pascal e C. Pioneira, 1999.
- [3] Adam Drozdek. Estrutura De Dados E Algoritmos Em C++. Cengage, 2010.