

# Atividade Prática 04

## “Manipulação de Árvores Binárias de Busca”

---

Universidade Tecnológica Federal do Paraná (UTFPR), campus Dois Vizinhos  
Curso de Engenharia de Software  
Disciplina de Estrutura de Dados - ED23S  
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

*Índices remissivos* facilitam muito a vida de quem vasculha por documentos longos. A diferença deste tipo para um índice convencional é que este indica a localização de uma palavra importante ou autor citado, e não um tópico ou subtítulo. Alguns livros e documentos utilizam índices remissivos enormes, com inúmeras indicações. Já outros economizam para não deixá-lo embaralhado. A figura 1 mostra um exemplo de índice remissivo encontrado em livros. No exemplo da figura, o índice remissivo de um livro lista alguns termos de destaque em ordem alfabética, seguidos das correspondentes páginas onde são encontrados.

Após experiências fracassadas em sua clínica médica, o professor M adquiriu uma editora. E como um bom ser nutrido do espírito d tecnologia, ele deseja criar índices remissivos automaticamente para qualquer livro que a editora for publicar. E mais uma vez, como confia em você, lhe selecionou para desenvolver esse projeto. Desta forma, explore e extrapole as

## Índice Remissivo

<b>A</b> Alef ..... 3 Análise ..... 1	<b>K</b> Kepler ..... 2	<b>Q</b> Quociente ..... 3
<b>B</b> Bola aberta ..... 3 fechada ..... 4	<b>L</b> Limite ..... 2 infinito ..... 4	<b>R</b> Razão ..... 3 Riemman ..... 4
<b>F</b> Função ..... 1	<b>M</b> Matemática ..... 2	<b>S</b> Somatório ..... 3
<b>H</b> História da Matemática 1	<b>N</b> Napier ..... 2	<b>T</b> Topologia ..... 3
	<b>P</b> Polinômios ..... 2	

Figura 1: Exemplo de índice remissivo de um livro. Os termos de destaque são listados em ordem alfabética, e as correspondentes páginas onde aparecem são apresentadas em sequência.

implementações de árvores binárias de busca para implementar um programa que crie um índice remissivo para um determinado texto de entrada.

## 2 Entradas do programa

O programa receberá dois arquivos texto como parâmetros de entrada:

- **arquivo de entrada:** um arquivo texto contendo o texto de entrada simulando um livro e o conteúdo de suas páginas (Figura 2a). A primeira linha do arquivo é uma tag `<termo:>` contendo todos os termos (palavras) que irão compor o índice remissivo:

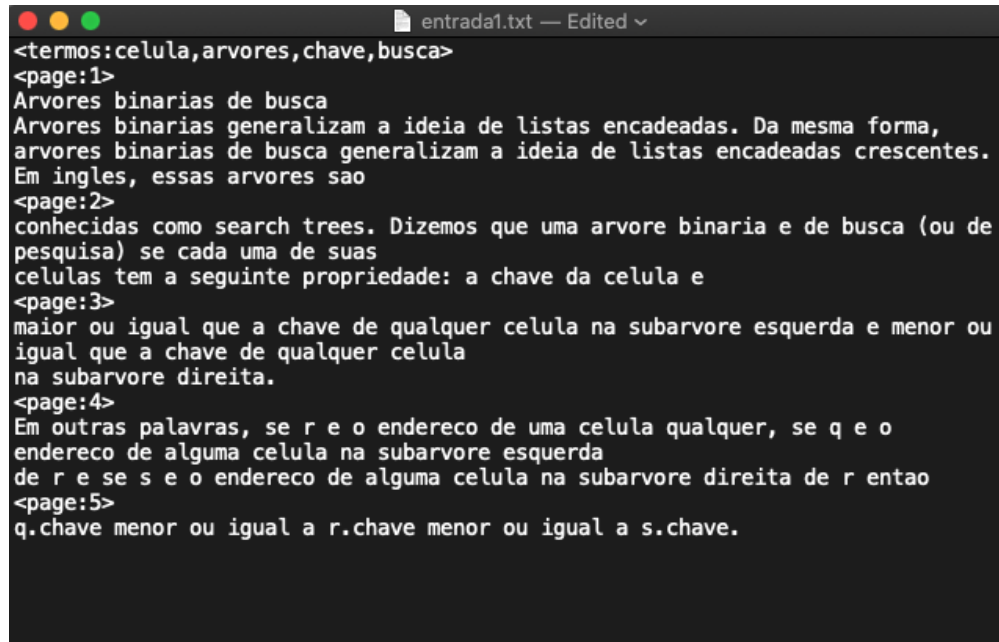
`<termos:palavra1,palavra2,palavra3,...palavraN>`

Na Figura 2a, os termos especificados pelo arquivo de entrada, e que serão consultados são:

`<termos:celula,arvores,chave,busca>`

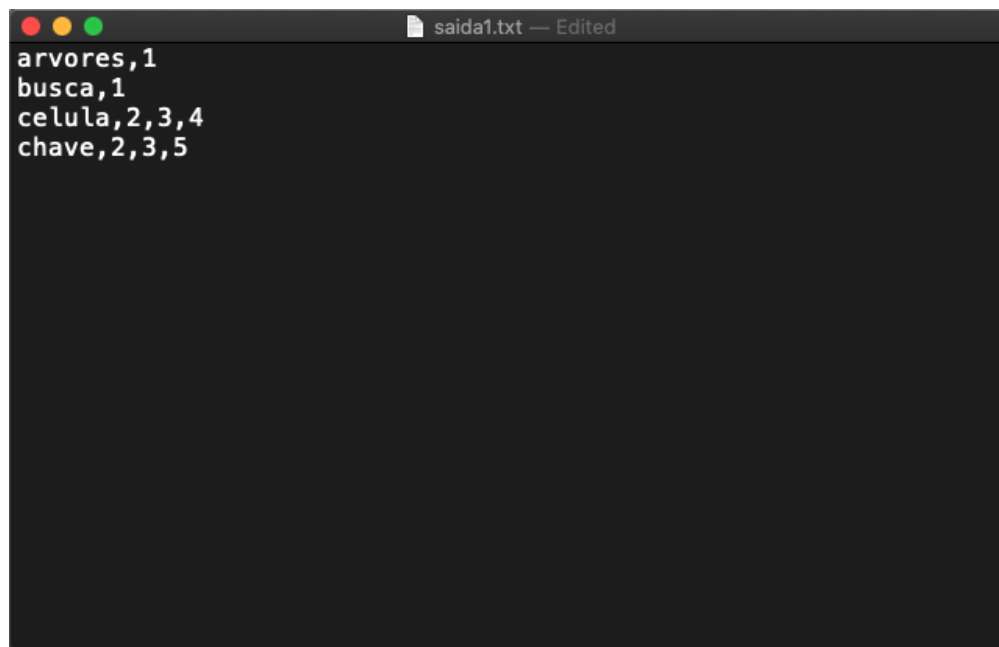
ou seja, as palavras **celula**, **arvores**, **chave** e **busca** serão verificadas em todo o texto (até o final do arquivo). O número de termos que podem compor o índice remissivo é variável, porém todos irão estar contidos dentro da tag para termos.

Da segunda linha em diante poderão haver diversas tags `<page:X>`, indicando o início de uma página fictícia do livro. Estas tags possuem também o correspondente número



```
<termos:celula,arvores,chave,busca>
<page:1>
Arvores binarias de busca
Arvores binarias generalizam a ideia de listas encadeadas. Da mesma forma,
arvores binarias de busca generalizam a ideia de listas encadeadas crescentes.
Em ingles, essas arvores sao
<page:2>
conhecidas como search trees. Dizemos que uma arvore binaria e de busca (ou de
pesquisa) se cada uma de suas
celulas tem a seguinte propriedade: a chave da celula e
<page:3>
maior ou igual que a chave de qualquer celula na subarvore esquerda e menor ou
igual que a chave de qualquer celula
na subarvore direita.
<page:4>
Em outras palavras, se r e o endereco de uma celula qualquer, se q e o
endereco de alguma celula na subarvore esquerda
de r e se s e o endereco de alguma celula na subarvore direita de r entao
<page:5>
q.chave menor ou igual a r.chave menor ou igual a s.chave.
```

(a) Exemplo de arquivo de entrada.



```
arvores,1
busca,1
celula,2,3,4
chave,2,3,5
```

(b) Exemplo de arquivo de saída.

Figura 2: Valores de entrada e correspondente arquivo de saída com índice remissivo gerado pelo programa.

de página. Por exemplo: `<page:1>` inicia a apresentação dos textos contidos na página 1; `<page:2>` os textos da página 2, e assim por diante. **Dica:** o texto que aparece entre duas tags de páginas deve ser manipulado como strings, e cada palavra manipulada isoladamente.

- **arquivo de saída:** é o arquivo texto onde serão impressos os correspondentes termos de busca, seguidos dos números de páginas onde foram encontrados, um termo por linha. Por exemplo, a palavra "chave" aparece nos textos das páginas 2, 3 e 4. Assim, deve-se imprimir no arquivo de saída (na Figura 2b) uma linha com essa informação:

celula, 2, 3, 4

No arquivo de saída, o índice remissivo deverá ser impresso em ordem alfabética.

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 `indice.c`:

`./indice 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 critério:

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 correta das 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 estruturas dinâmicas, 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.

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

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

Exemplo:

```
ED1-2021-2-AT04-IndiceRemissivo-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](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.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.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>

## Referências

- [1] Thomas H. Cormen,; Ronald Rivest; Charles E. Leiserson; Clifford Stein. Algoritmos - Teoria e Prática - 3<sup>a</sup> 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.