

# Atividade Prática 04

## Estruturas de Índices

---

Universidade Tecnológica Federal do Paraná (UTFPR), campus Apucarana  
Curso de Engenharia de Computação  
Disciplina de Estrutura de Dados 2 - EDCO4B  
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

Depois de alguns semestres tendo aula com o professor M vocês perceberam que ele é uma pessoa bem normal. E como uma pessoa normal, muitas vezes seu humor varia ao longo dos dias: há dias mais desanimados e outros mais animados. Porém, uma estratégia para sempre tentar abstrair e ficar bem é ouvir música. É comum o professor M colocar um som de fundo nas suas aulas para que tanto ele, como os alunos, possam se distrair um pouco.

E como todo viciado em música, professor M gosta de anotar/registrar as músicas que são de seu gosto. Depois de ouvir por um tempo rádio ou *players* de música, ele corre anotar as informações das músicas que ouviu e curtiu. Para cada música ele anota:

- **ano:** o ano que a música foi lançada;
- **duração:** a duração em minutos e segundos;
- **título:** o título da música;
- **artista:** o artista que gravou a música;

- **gênero:** o gênero da música; e
- **idioma:** o idioma da canção.

Depois de um tempo, ele conseguiu criar um arquivo bacana com uma coleção de músicas variadas. Um vislumbre do arquivo de músicas do professor M pode ser visto na Figura 1.

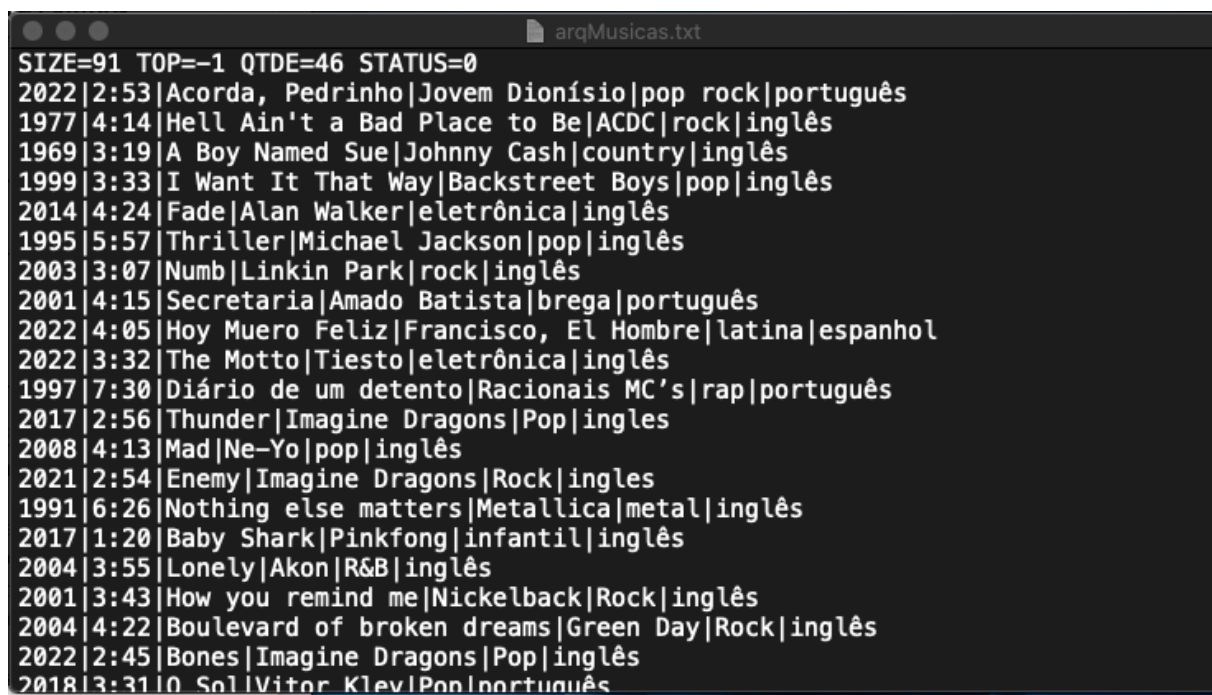


Figura 1: Arquivo de Músicas do professor M.

Pensando em melhorar o seu processo de consulta às músicas, o professor M pediu para vocês desenvolverem índices secundários que permitam consultas dos valores existentes no arquivo. Por exemplo: retornar todas as músicas de um artista específico, retornar todas as músicas de um gênero, ou ainda retornam todas as músicas que são cantadas em português. Essa é sua nova **missão**: desenvolver um programa com índices secundários que satisfaça as consultas de músicas do arquivo do professor M.

## 2 Entradas do programa

O programa receberá **três** arquivos texto como parâmetros: um arquivo de dados com as músicas a serem manipuladas, um arquivo de consulta, e um arquivo de saída. Abaixo, iremos detalhar cada um deles.

### 2.1 Arquivo de dados

O primeiro parâmetro é o arquivo de dados, o mesmo que é apresentado na Figura 1. Ele lista todas as músicas preferidas do professor M. O armazenamento das músicas é feito em

um arquivo de registros de tamanho fixo, mas de campos com tamanho variáveis. Cada campo é separado por um pipe ( | ), e os registros finalizados por uma quebra de linha. Há um registro de cabeçalho (*header*) que contém algumas informações importantes para a execução do programa. Essas informações estão sumarizadas na Tabela 1. Uma sugestão de estrutura para codificar um registro (músicas) é apresentada na Figura 2.

```
// Estrutura de Musica
typedef struct {
    char ano[5];      // no padrão de 4 dígitos XXXX
    char duracao[6];  // no padrão MM:SS
    char titulo[31];
    char artista[21];
    char genero[12];
    char idioma[12];
} Musica;
```

Figura 2: Estrutura de música a se manipulada na aplicação.

Tabela 1: Parâmetros contidos no cabeçalho do arquivo de entrada

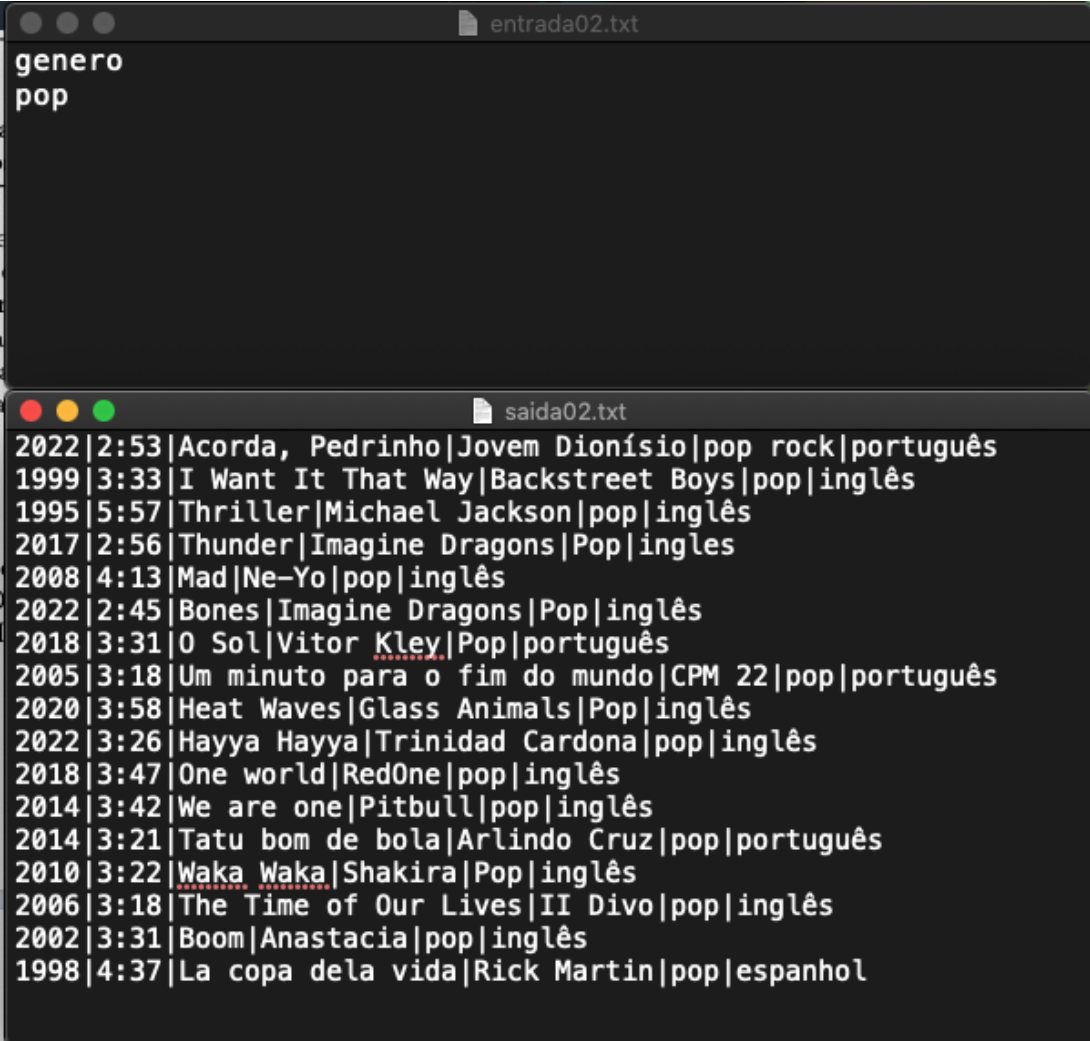
Parâmetro	Descrição	Opções Válidas
SIZE	tamanho dos registros (bytes/caracteres)	91
TOP	índice da posição do topo da pilha lógica de reuso	-1
QTDE	quantidade de registros contidos no arquivo de entrada	[0, ...]
STATUS	indica se os índices encontram-se salvos nos arquivos de dump	0

No parâmetro **SIZE**, os registros terão valores fixos de 91 caracteres, contando os pipes e quebra de linha. O parâmetro **TOP** controla o reuso dos registros, depois de operações de leitura e escrita. Nessa atividade o valor será constante em -1, pois não iremos modificar os arquivos, apenas consultá-los. O parâmetro **QTDE** contém a quantidade de registros armazenados no arquivo. O último parâmetro (**STATUS**), indica se o índice, após ser manipulado na memória, está salvo nos arquivos de *dump* (backup). Na nossa aplicação, esse valor não será modificado.

## 2.2 Arquivo de Consulta

Um arquivo texto contendo duas informações (tipo de índice a ser criado e a *string* de busca), uma por linha. Assim, na primeira linha existirá o nome do campo ao qual criaremos o índice secundário. As opções válida são: ano, título, artista, gênero e idioma. Caso o arquivo de consulta possua uma string na primeira linha diferente dos valores acima descritos, o programa deve indicar o erro e não executar.

Na segunda linha do arquivo teremos a *string* de busca (o valor de consulta). Um exemplo é apresentado na Figura 3. No exemplo em questão, queremos criar um índice secundário



```
genero
pop

2022|2:53|Acorda, Pedrinho|Jovem Dionísio|pop rock|português
1999|3:33|I Want It That Way|Backstreet Boys|pop|inglês
1995|5:57|Thriller|Michael Jackson|pop|inglês
2017|2:56|Thunder|Imagine Dragons|Pop|ingles
2008|4:13|Mad|Ne-Yo|pop|inglês
2022|2:45|Bones|Imagine Dragons|Pop|inglês
2018|3:31|0 Sol|Vitor Kley|Pop|português
2005|3:18|Um minuto para o fim do mundo|CPM 22|pop|português
2020|3:58|Heat Waves|Glass Animals|Pop|inglês
2022|3:26|Hayya Hayya|Trinidad Cardona|pop|inglês
2018|3:47|One world|RedOne|pop|inglês
2014|3:42|We are one|Pitbull|pop|inglês
2014|3:21|Tatu bom de bola|Arlindo Cruz|pop|português
2010|3:22|Waka Waka|Shakira|Pop|inglês
2006|3:18|The Time of Our Lives|II Divo|pop|inglês
2002|3:31|Boom|Anastacia|pop|inglês
1998|4:37|La copa dela vida|Rick Martin|pop|espanhol
```

Figura 3: Valores de entrada e correspondente arquivo de saída gerado pelo programa.

com as informações dos gêneros das músicas, e retornar todas aquelas que são músicas de pop.

## 2.3 Arquivo de saída

Um arquivo texto contendo a busca realizada pelo programa após criar o correspondente índice secundário. A Figura 3 mostra um exemplo do arquivo de saída onde são retornadas todas as músicas do gênero pop contidas no arquivo de dados (base de dados). lembre-se que no arquivo de saída, devem existir mensagens de erro, ou indicativos de que a consulta não pode ser executada (nenhum valor encontrado).

### 3 Rodando o 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 dados] [arquivo de entrada] [arquivo de saída]
```

Exemplo de execução de um programa chamado `indice.c`:

```
./indice musicas.txt entrada01.txt saida01.txt
```

### 4 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;
- arquivos de entrada vazio (sem informação);
- arquivos de entrada fora do padrão esperado (opções inválidas para consulta);
- etc.

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

### 5 Critérios 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 a leitura dos dados de entrada via arquivo texto;
7. implementação correta das estruturas necessárias (campos, registros e sua manipulação, ordenação das chaves);
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.

## 6 Padrão de nomenclatura

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

```
ED2-<ANO>-<SEMESTRE>-AT04-IndiceSecundario-<NOME>.c
```

Exemplo:

```
ED2-2022-1-AT04-IndiceSecundario-RafaelMantovani.c
```

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

## 7 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>
- Arquivos em Python:
  - <https://www.geeksforgeeks.org/reading-writing-text-files-python/>
  - [https://www.w3schools.com/python/python\\_file\\_open.asp](https://www.w3schools.com/python/python_file_open.asp)
  - <https://www.pythontutorial.net/python-basics/python-read-text-file/>
- Argumentos de Linha de comando em C(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>
- Argumentos de Linha de comando no Python:
  - [https://www.tutorialspoint.com/python3/python\\_command\\_line\\_arguments.htm](https://www.tutorialspoint.com/python3/python_command_line_arguments.htm)
  - <https://realpython.com/python-command-line-arguments/>
  - <http://devfuria.com.br/python/sys-argv/>

## Referências

- [1] Michael J. Folk; Bill Zoellick; Greg Riccardi. File Structures, 3rd edition, Addison-Wesley, 1997.
- [2] Thomas H. Cormen,; Ronald Rivest; Charles E. Leiserson; Clifford Stein. Algoritmos - Teoria e Prática - 3ª Ed. Elsevier - Campus, 2012.
- [3] Nivio Ziviani. Projeto de algoritmos com implementações: em Pascal e C. Pioneira, 1999.
- [4] Adam Drozdek. Estrutura De Dados e Algoritmos em C++. Cengage, 2010.