Universidade Federal de São Carlos
Bacharelado em Ciência da Computação
Estruturas de Dados II
Prof. Tiago A. Almeida <talmeida@ufscar.br>

TRABALHO 02 - ÁRVORES-B

Atenção

- **Prazo de entrega: 15/11/2017 – 23h55 (via Moodle).**

O sistema de cadastro de jogos da *Steam* foi um sucesso e logo diversos concorrentes decidiram aderir ao sistema. Contudo, com o crescimento acelerado do arquivo de dados, as consultas começaram a ficar lentas e, em consequência, seu programa vem perdendo credibilidade e recebendo uma enxurrada de reclamações dos usuários.

Após analisar o cenário atual, concluiu-se que o uso de índices simples não é mais viável para realizar buscas no arquivo de dados, e que a melhor saída é usar índices de árvores-B para aumentar a eficiência do sistema.

Lembrando, cada jogo (registro no arquivo de dados) é composto pelos seguintes campos:

- *Código*: composição de letras maiúsculas das duas primeiras letras do nome do jogo, seguido das duas primeiras letras do nome da desenvolvedora, do dia e mês do lançamento (com dois dígitos cada) e da classificação etária do jogo (dois dígitos), ex: `ROPS070700`. Esse campo é a *chave primária*, portanto, não poderá existir outro valor idêntico na base de dados;
- *Nome do jogo* (título pelo o qual os jogadores conhecem o jogo, ex: `Rocket League`);
- *Desenvolvedora* (nome da empresa que produziu o título, ex: `Psyonix`);
- *Data de lançamento* (data no formato `DD/MM/AAAA`, ex: 07/07/2015);
- *Classificação etária* (inteiro com 2 dígitos, representando a faixa etária recomendada, utilize zero quando um título não houver restrições de conteúdo para nenhuma faixa etária, ex: `00`);
- *Preço-base* (valor de ponto flutuante com precisão de dois dígitos referente ao preço base do produto sem descontos, ex: `36.99`);
- *Desconto* (inteiro com 3 dígitos, contendo a porcentagem de desconto que será abatida no preço original do título durante a temporada de vendas, ex: `040` - nesse caso o título em questão ficaria com o preço final igual a 22.19.
- *Categorias* (campo multi-valorado separado pelo caractere '|' se houver mais de uma categoria, ex: `ACAO|CORRIDA|ESPORTES|INDIE`);

Garantidamente, nenhum campo de texto receberá caractere acentuado.

Tarefa

Desenvolva um programa que permita ao usuário manter uma base de dados de jogos. O programa deverá permitir:

1. Inserir um novo jogo;
2. Modificar o campo **desconto** de um jogo a partir da chave primária;
3. Buscar títulos a partir:
   1) da chave primária
   2) desenvolvedora e nome do jogo (índice secundário).
4. Listar todos os jogos da base de dados ordenados por:
   1) impressão pré-ordem da árvore-B que representa o índice primário
   2) impressão pré-ordem da árvore-B que representa o índice secundário
5. Visualizar o arquivo de dados;
6. Visualizar o arquivo de índice primário;
7. Visualizar o arquivo de índice secundário.

Novamente, nenhum arquivo ficará salvo em disco. O arquivo de dados e os dois de índices serão simulados em *strings*, sendo que os índices deverão ser sempre criados na inicialização do programa e manipulados em memória RAM até o término da execução. Suponha que há espaço suficiente em memória RAM para todas as operações.

Arquivo de dados

Como este trabalho será corrigido pelo `OnlineJudge` e o sistema não aceita funções que manipulam arquivos, os registros serão armazenados e manipulados em uma *string* que simula o arquivo aberto. Você deverá utilizar as variáveis globais `ARQUIVO`, `ARQUIVO_IP` e `ARQUIVO_IS` e funções de leitura e escrita em *strings*, como `sprintf` e `sscanf`, para simular as operações de leitura e escrita em arquivo.

O arquivo de dados deve ser ASCII (arquivo texto), organizado em registros de tamanho fixo de 192 bytes. Os campos *título do jogo*, *nome da desenvolvedora* e *categorias* devem ser de tamanho variável. Os demais campos devem ser de tamanho fixo: *código* (10 bytes), *data de lançamento* (10 bytes), *preço-base* (7 bytes), *classificação etária* (2 bytes) e *desconto* (3 bytes). A soma de bytes dos campos fornecidos (incluindo os delimitadores necessários) nunca poderá ultrapassar 192 bytes. Os campos do registro devem ser separados pelo caractere delimitador @ (arroba). Cada registro terá 7 delimitadores, mais 32 bytes ocupados pelos campos de tamanho fixo. Você precisará garantir que os demais campos juntos ocupem um máximo de 153 bytes. Caso o registro tenha menos de 192 bytes, o espaço adicional deve ser marcado com o caractere # de forma a completar os 192 bytes. Para evitar que o registro exceda 192 bytes, os campos variáveis devem ocupar no máximo 51 bytes.

```
LEWA041200@LEGO JURASSIC WORLD@WARNER BROS@04/12/2006@00@0055.5
5@079@ACTION|ADVENTURE|CASUAL|MULTIPLAYER|RPG|RACING@##########
##########################################################
###MEKO140118@METAL GEAR SOLID V: THE PHANTOM PAIN@KOJIMA PRODU
CTIONS@14/01/2010@18@0192.25@033@ACTION|ADVENTURE|CASUAL|INDIE|
MULTIPLAYER|RPG@#################################################
######BO2K241103@BORDERLANDS 2@2K GAMES@24/11/2014@03@0055.92@1
00@ACTION|ADVENTURE|INDIE|MULTIPLAYER|RPG|RACING@###############
##########################################################
#########THED271000@THE BINDING OF ISAAC@EDMUND MCMILLEN@27/10/
2015@00@0037.65@023@ACTION|ADVENTURE|INDIE|MULTIPLAYER|RPG|RACI
NG@##########################################################
############HAVA160314@HALF-LIFE 2@VALVE@16/03/2003@14@0062.25@
033@MULTIPLAYER@#################################################
##########################################################
###############
```

Note que não há quebras de linhas no arquivo (elas foram inseridas aqui apenas para exemplificar a sequência de registros).

Instruções para as operações com os registros:

- **Inserção:** cada jogo deverá ser inserido no final do arquivo de dados e atualizado nos índices.
- **Atualização:** o único campo alterável é o de *Desconto*. O registro deverá ser localizado acessando o índice primário e o desconto deverá ser atualizado no registro na mesma posição em que está (não deve ser feita remoção seguida de inserção). Note que o campo de *Desconto* sempre terá 3 dígitos.

Arquivo de Índices

No cenário atual, **os índices não cabem em RAM** e, portanto, para simular essa situação, dois "*Arquivos de Índices*: `iprimary` e `idev`" deverão ser criados na inicialização do programa e manipulados em RAM até o encerramento da aplicação.

Para ambas as árvores, as ordens serão informadas pelo usuário e **a promoção deverá ser sempre pelo sucessor imediato** (menor chave da sub-árvore direita). Todo novo nó criado deverá ser inserido no final do respectivo arquivo de índice.

1. `iprimary`: índice primário (Árvore-B), contendo as chaves primárias e os RRNs dos registros no arquivo de dados. Cada registro da árvore primária é composto por:
   - 3 bytes para a quantidade de chaves;
   - (Ordem -1) * (10 bytes de chave primária + 4 bytes para o RRN do arquivo de dados). Para as chaves não usadas, preencha todos os bytes com `'#'`;
   - 1 byte para indicar se o nó é folha `'F'` ou não `'N'`;
   - Por fim, uma quantia de (ordem * 3 bytes) para indicar os RRNs dos nós descendentes do nó atual. Note que esse RRN se refere ao próprio arquivo de índice primário. Utilize `'***'` para indicar que aquela posição do vetor de descendentes é nula.

Exemplo da representação da árvore B de ordem 3 após a inserção das chaves iniciadas com: LEWA, MEKO e BO2K (nesta ordem).

Figura 1: Árvore-B de ordem 3 após a inserção do primeiro registro

Em disco, o arquivo de índice primário seria:

```
001LEWA0412000000##############F*********
```

Figura 2: Árvore-B de ordem 3 após a inserção do segundo registro

Em disco:

```
002LEWA0412000000MEKO1401180001F*********
```

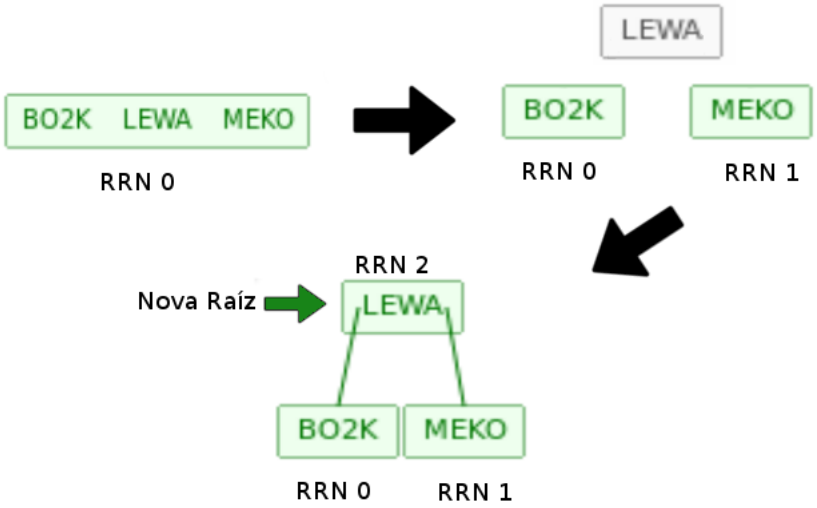

Figura 3: Árvore-B de ordem 3 após a inserção do terceiro registro

Note que ao inserir a chave `BO2K` ocorreu um *overflow* na raiz, sendo necessário criar então, 2 novos nós (de RRN 1 e 2 respectivamente). O primeiro será para a redistribuição de chaves, e o segundo para receber a promoção de chave que será a nova raiz. Assim, o arquivo de índice primário resultante será:

```
001BO2K2411030002##############F*********
001MEKO1401180001##############F*********
001LEWA0412000000##############N000001***
```

Note que, aqui também não há quebras de linhas no arquivo. Elas foram inseridas apenas para exemplificar a sequência de registros.

2. `idev`: índice secundário (Árvore-B) contendo chaves compostas por: (*i*) a chave primária do respectivo registro e (*ii*) o nome da desenvolvedora concatenado com '`$`' e o nome do jogo, sendo que as chaves devem ser ordenadas pela ordem lexicográfica da segunda *string* (*ii*). Cada registro da árvore secundária é composto por:

   - 3 bytes para a quantidade de chaves;
   - (Ordem -1) * (10 bytes de chave primária + 101 bytes da *string*). Para as chaves não usadas, preencha todos os 111 bytes com '`#`';
   - 1 byte para indicar se o nó é uma folha '`F`' ou não '`N`'; e
   - por fim, uma quantia de (ordem * 3 bytes) para indicar os RRNs dos nós descendentes. Note que assim como no índice primário, esse RRN se refere ao próprio arquivo de índice secundário. Utilize '`***`' para indicar que aquela posição do vetor de descendentes é `NULL`.

   Exemplo de arquivo de índice secundário representado por uma árvore-B de ordem 4, após a inserção das chaves na ordem: '`WARNER BROS$LEGO JURASSIC WORLD`' (`WALE`), '`KOJIMA PRODUCTIONS$METAL GEAR`' (`KOME`) e '`2K GAMES$BORDERLANDS`' (`2KBO`).

Figura 4: Índice secundário estruturado em Árvore-B de ordem 4.

Em disco, o arquivo de índice secundário seria:

```
003BO2K2411032K GAMES$BORDERLANDS 2############################
##################################################MEKO140118KO
JIMA PRODUCTIONS$METAL GEAR SOLID V: THE PHANTOM PAIN##########
#####################################LEWA041200WARNER BROS$LEGO
JURASSIC WORLD#####################################################
####################F************
```

Novamente, não há quebras de linhas no arquivo. Elas foram inseridas apenas para exemplificar a sequência de registros.

É terminantemente proibido manter uma cópia dos índices inteiros em TADs. A única informação que você deverá manter todo tempo em memória, é o RRN da raiz de cada árvore. Assuma que um nó do índice corresponde a uma página e, portanto, cabe no *buffer* de memória. Dessa forma, carregue apenas um nó da Árvore-B por vez.

Deverá ser desenvolvida uma rotina para a criação de cada índice. Os índices serão criados e manipulados sempre utilizando os pseudo-arquivos de índices. Note que o ideal é que a árvore-B `iprimary` seja a primeira a ser criada.

Para que isso funcione corretamente, o programa, ao iniciar precisa realizar os seguintes passos:

1. Perguntar ao usuário se ele deseja informar um arquivo de dados:
   - Se sim: recebe o arquivo inteiro e armazena no vetor `ARQUIVO`.
   - Se não: considere que o arquivo está vazio.
2. Inicializar as estruturas de dados dos índices:
   - Solicitar as ordens m e n das duas árvores e criar a estrutura dos índices.

Interação com o usuário

O programa deve permitir interação com o usuário pelo console/terminal (modo texto) via menu.

A primeira pergunta do sistema deverá ser pela existência ou não do arquivo de dados. Se existir, deve ler o arquivo e armazenar no vetor `ARQUIVO`. Em seguida, o sistema deverá perguntar pelas ordens m e n das árvores-B usadas para indexação.

As seguintes operações devem ser fornecidas (nessa ordem):

1. **Cadastro.** O usuário deve ser capaz de inserir um novo jogo. Seu programa deve ler os seguintes campos (nessa ordem): **nome do jogo**, **desenvolvedora**, **data de lançamento**, **classificação etária**, **preço-base**, **desconto** e **categorias**. Note que a chave **não é** inserida pelo usuário, você precisa gerar a chave para gravá-la no arquivo de dados e nos índices. Garantidamente, os campos serão fornecidos de maneira regular, não sendo necessário um pré-processamento da entrada, exceto na opção de **Alteração**. Se um novo registro possuir a chave gerada igual a de um outro registro já presente no arquivo de dados, a seguinte mensagem de erro deverá ser impressa: "`ERRO: Já existe um registro com a chave primária AAAA999999.\n`", onde `AAAA999999` corresponde à chave primária do registro que está sendo inserido e `\n` indica um pulo de linha após a impressão da frase.
2. **Alteração.** O usuário deve ser capaz de alterar o desconto de um jogo informando a sua chave primária. Caso ele não exista, seu programa deverá exibir a mensagem "`Registro não encontrado!\n`" e retornar ao menu. Caso o registro seja encontrado, certifique-se de que o novo valor informado está dentro dos padrões (*i.e.*, 3 bytes, com o valor entre 000 e 100) e, nesse caso, altere o valor do campo diretamente no arquivo de dados. Caso contrário, exiba a mensagem "`Campo inválido!\n`" e solicite a digitação novamente.
3. **Busca.** O usuário deve ser capaz de buscar por um jogo:
   - **1. por código:** solicitar ao usuário a chave primária. Caso o jogo não exista, seu programa deve exibir a mensagem "`Registro nao encontrado!`" e retornar ao menu principal. Caso o jogo exista, todos os dados deverão ser impressos na tela de forma formatada, exibindo os campos na mesma ordem de inserção. Em ambos os casos, seu

programa deverá imprimir o caminho percorrido na busca exibindo as chaves contidas nos nós percorridos. **Na última linha do caminho percorrido, adicione uma quebra de linha adicional.**

Por exemplo, considere a seguinte árvore-B de ordem 3 resultante da inserção das seguintes chaves: `LEKO041200`, `MEVA140118`, `BOBE241103`, `THSE271000`, `HAVA160314`, `CABE180614`, `COUB011221`, `GRNA120803`, `BASQ240318` e `XCBE201105` (Figura 5).

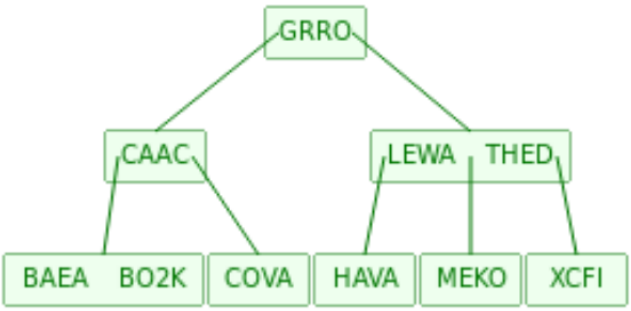

Figura 5: Índice primário estruturado em Árvore-B de ordem 3.

A busca pela chave `MEKO140118` e `CAAC999999` retornará:

```
**********BUSCAR**********
Busca por MEKO140118. Nos percorridos:
GRRO120803
LEWA041200, THED271000
MEKO140118

MEKO140118
METAL GEAR SOLID V: THE PHANTOM PAIN
KOJIMA PRODUCTIONS
14/01/2010
18
0128.80
ACTION ADVENTURE CASUAL INDIE MULTIPLAYER RPG
**********BUSCAR**********
Busca por CAAC999999. Nos percorridos:
GRRO120803
CAAC180614
COVA011221

Registro(s) nao encontrado!
```

- **2. por desenvolvedora e nome do jogo:** solicitar ao usuário o nome da desenvolvedora seguido do nome do jogo. Caso nenhum jogo tenha sido encontrado, o programa deve exibir a mensagem "`Registro(s) nao encontrado!`" e retornar ao menu principal. Exemplo de utilização para as buscas: {'`2K GAMES`','`BORDERLANDS 2`'} e {'`VALVE`','`HALF-LIFE 3`'}

```
**********BUSCAR**********
Busca por 2K GAMES$BORDERLANDS 2.
Nos percorridos:
EA GAMES$BATTLEFIELD 4, KOJIMA PRODUCTIONS$METAL GEAR SOLID V: THE PHANTOM PAIN,
2K GAMES$BORDERLANDS 2, ACTIVISION$CALL OF DUTY MODERN WARFARE III

BO2K241103
BORDERLANDS 2
2K GAMES
24/11/2014
03
0000.00
ACTION ADVENTURE INDIE MULTIPLAYER RPG RACING
**********BUSCAR**********
Busca por VALVE$HALF-LIFE 3.
Nos percorridos:
EA GAMES$BATTLEFIELD 4, KOJIMA PRODUCTIONS$METAL GEAR SOLID V: THE PHANTOM PAIN,
    VALVE$HALF-LIFE 2

Registro(s) nao encontrado!
```

4. **Listagem.** O sistema deverá oferecer as seguintes opções de listagem:

- **1. árvore-B primária:** imprime a `iprimary` (somente o campo de chave primária) usando varredura **pré-ordem**. Caso não haja nenhum registro, imprima a mensagem de '`Registro(s) nao encontrado!`' Imprimir um nó por linha, começando pelo nível da árvore em que se encontra o nó raiz (nível 1) seguido da chave. Por exemplo, considere a árvore-B apresentada na Figura 5, a sua listagem resultaria em:

```
**********LISTAR**********
1 - GRRO120803
2 - CAAC180614
3 - BAEA240318, BO2K241103
3 - COVA011221
2 - LEWA041200, THED271000
3 - HAVA160314
3 - MEKO140118
3 - XCFI201105
```

- **2. Árvore-B secundária:** realiza uma travessia **em ordem**, exibindo todos os títulos de jogos na ordem lexicográfica da desenvolvedora e título. Exemplo:

```
*******************************LISTAR*********************************
2K GAMES------------------------ BORDERLANDS 2----------------------
ACTIVISION---------------------- CALL OF DUTY MODERN WARFARE III-----
EA GAMES------------------------ BATTLEFIELD 4----------------------
EDMUND MCMILLEN----------------- THE BINDING OF ISAAC---------------
FIRAXIS GAMES------------------- XCOM 2-----------------------------
KOJIMA PRODUCTIONS-------------- METAL GEAR SOLID V: THE PHANTOM PAIN
ROCKSTAR GAMES------------------ GRAND THIEF AUTO V-----------------
VALVE--------------------------- COUNTER STRIKE: GLOBAL OFFENSIVE----
VALVE--------------------------- HALF-LIFE 2------------------------
```

  Note que o número de tracejado foi diminuído devido ao tamanho da margem do PDF, porém, cada linha é composta por: 50 caracteres (desenvolvedora) + 1 espaço em branco + 50 caracteres (título) + retorno de linha.

5. **Imprimir Arquivo de dados.** Imprime o Arquivos de dados, caso esteja vazio apresente a mensagem "`Arquivo vazio!`".

6. **Imprimir Índice Primário.** Imprime o Arquivo de índice primário, caso esteja vazio apresente a mensagem "`Arquivo vazio!`".

7. **Imprimir Secundário.** Imprime o Arquivo de índice secundário, caso esteja vazio apresente a mensagem "`Arquivo vazio!`".

8. **Finalizar.** Encerra a execução do programa. Ao final da execução, libere toda a memória alocada pelo programa caso ainda possua alguma.

Implementação

Implemente suas funções utilizando o código-base fornecido. Dessa vez, **não é permitido modificar os trechos de código pronto ou as estruturas já definidas**. Ao imprimir alguma informação para o usuário, utilize as constantes definidas. Ao imprimir um registro, utilize a função `exibir_registro()`.

Tenha atenção redobrada ao implementar as operações de busca e listagem da árvore-B. Atente-se às quebras de linhas requeridas e não adicione espaços em branco após o último caractere imprimível. A saída deverá ser exata para não dar conflito com o `OnlineJudge`. Em caso de dúvidas, examine os casos de teste.

Você deve criar obrigatoriamente as seguintes funcionalidades:

- Criar o índice primário: deve construir o índice primário a partir do arquivo de dados e da ordem m informada na inicialização do programa;

- Criar o índice secundário: deve construir o índice secundário a partir do arquivo de dados e da ordem n informada na inicialização do programa;
- Inserir um registro: modificar o arquivo de dados e os arquivos de índices.
- Buscar por registros: buscar por registros pela chave primária ou secundária.
- Alterar um registro: modificar o arquivo de dados.
- Listar registros: listar as árvores-B.
- Finalizar: deverá ser chamada ao encerrar o programa e liberar toda a memória alocada.

Utilizar a linguagem `ANSI C`.

Dicas

- Você nunca deve perder a referência do começo dos arquivos, então não é recomendável percorrer as *strings* diretamente pelos ponteiros `ARQUIVO`, `ARQUIVO_IP` e `ARQUIVO_IS`. Um comando equivalente a `fseek(f, 192, SEEK_SET)` é `char *p = ARQUIVO + 192`.
- Diferentemente do `fscanf`, o `sscanf` não movimenta automaticamente o ponteiro após a leitura.
- O `sprintf` adiciona automaticamente o caractere `\0` no final da *string* escrita. Em alguns casos você precisará sobrescrever a posição manualmente. Você também pode utilizar o comando `strncpy` para escrever em *strings*, esse comando, diferentemente do `sprintf`, não adiciona o caractere nulo no final.
- Ao utilizar o comando `strcpy` certifique-se que a *string* destinatária possui tamanho maior ou igual que a de origem, caso contrário poderá realizar escrita em espaço inapropriado da memória. Como alternativa use a `strncpy`.
- Não é possível retornar mais de um valor diretamente em C, mas a linguagem disponibiliza a criação de `structs` e também a passagem por referência para simular tal recurso.
- A função `strtok` permite navegar nas *substrings* de uma certa *string* dado(s) o(s) delimitador(es). Porém, tenha em mente que ela deve ser usada em uma cópia da *string* original, pois ela modifica o primeiro argumento.
- Utilize ferramentas de depuração, tais como GDB e Valgrind, para encontrar erros específicos. A fim de aumentar sua produtividade.

CUIDADOS

Leia atentamente os itens a seguir.

1. O projeto deverá ser submetido no `OnlineJudge` em um único arquivo com o nome `{RA}_ED2_T02.c`, sendo `{RA}` correspondente ao número do seu RA;

2. Não utilize acentos nos nomes de arquivos;

3. Dúvidas conceituais deverão ser colocadas nos horários de atendimento. Dificuldades em implementação, consultar o monitor da disciplina nos horários estabelecidos;

4. **Documentação:** inclua cabeçalho, comentários e indentação no programa;

5. **Erros de compilação:** nota **zero** no trabalho;

6. **Tentativa de fraude:** nota **zero na média** para todos os envolvidos.