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: 26/10/2016 – 23h55 (via Moodle).**
  Após o prazo, o trabalho não será considerado!

Indexação usando Árvores-B

O sistema de cadastro de *Pokémons* foi um sucesso e logo diversos jogadores de *Pokémon GO* decidiram utilizar o 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 *Pokémon* tem os seguintes campos, nesta ordem, armazenados no arquivo de dados:

- *Código* (composição de letras maiúsculas da primeira letra do nome da equipe, seguida da primeira letra do nome do treinador, seguida das duas primeiras letras do nome do *Pokémon* e do dia, mês, hora e minuto da captura (com dois dígitos cada)). Ex: MABU13081125 - esse campo é a chave primária, portanto, não poderá existir outro valor idêntico na base de dados;
- *Nome do Pokémon* (nome da espécie de Pokémon. Ex: Bulbasaur);
- *Tipo(s) do Pokémon* (separados por uma barra '/', caso o *Pokémon* se enquadre em mais de um. Ex: Grama/Venenoso);
- *Combat Points – CP* (pontos de combate do *Pokémon*, com 4 dígitos inteiros e 2 dígitos decimais, separados por um ponto '.'. Ex: 1013.06);
- *Data de captura* (data de captura do *Pokémon*, no formato DD/MM/AA. Ex: 13/08/16);
- *Hora de captura* (hora da captura do *Pokémon*, no formato HH:MM. Ex: 11:25);
- *Nome do treinador* (nome do treinador que capturou o *Pokémon*. Ex: Ash);

- *Nível do treinador* (nível do treinador que capturou o *Pokémon*, com 3 dígitos inteiros. Ex: 095);
- *Nome da equipe* (nome da equipe do treinador que capturou o *Pokémon*. Ex: Mystic);

Garantidamente, nenhum campo de texto receberá caractere acentuado.

Tarefa

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

1. Inserir um novo *Pokémon*;
2. Modificar o campo **combat points** de um *Pokémon* a partir de sua chave primária;
3. Buscar *Pokémons* a partir:
   1) de sua chave primária,
   2) do nome do *Pokémon*, ou
   3) do nome da equipe a qual o *Pokémon* pertence.
4. Listar todos os *Pokémons* da base, ordenados por:
   1) impressão pré-ordem da árvore-B,
   2) nome do *Pokémon* (ordem lexicográfica), ou
   3) nome da equipe a qual o *Pokémon* pertence.

Dessa vez, nenhum arquivo ficará salvo em disco. O arquivo de dados será simulado em uma *string* e os índices serão 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 deve utilizar a variável global `ARQUIVO` e funções de leitura e escrita em *strings*, como `sprintf` e `sscanf`, para simular as operações de leitura e escrita em arquivo.

Dicas

- Você nunca deve perder a referência do começo do arquivo, então não é recomendável percorrer a *string* diretamente pelo ponteiro `ARQUIVO`. 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.

O arquivo de dados deve ser organizado em registros de tamanho fixo de 192 *bytes*. Os campos que possuam letras em sua composição devem ser convertidos para que todas elas sejam maiúsculas no arquivo de dados. Os campos `nome do Pokémon`, `tipo do Pokémon`, `nome do treinador` e `nome da equipe` devem ser de tamanho variável. Os demais campos devem ser de tamanho fixo: `combat points` (7 *bytes*), `data de captura` (8 *bytes*), `hora da captura` (5 *bytes*), `nível do treinador` (3 *bytes*) e `código` (chave primária – 12 *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 9 delimitadores, mais 35 *bytes* ocupados pelos campos de tamanho fixo. Você precisará garantir que os demais campos juntos ocupem um máximo de 148 *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*, cada campo de tamanho variável deve ocupar no máximo 37 *bytes*. O programa deve impedir a inserção de registros com campos de mais de 37 *bytes*.

```
MABU13081125@BULBASAUR@GRAMA/VENENOSO@1013.06@13/08/16@11:25@ASH
@095@MYSTIC@###########################################################
############################################################################
IBGE10081309@GEODUDE@PEDRA/TERRA@0808.00@10/08/16@13:09@BROCK@08
8@INSTINCT@###############################################################
############################################################################
VMPS26082211@PSYDUCK@AGUA@1010.55@26/07/16@22:11@MISTY@093@VALOR
@#########################################################################
############################################################################
MAPI08080500@PIKACHU@ELETRICO@0876.54@08/08/16@05:00@ASH@095@MYS
TIC@#######################################################################
############################################################################
MACH02090259@CHARIZARD@FOGO/VOADOR@2222.22@02/09/16@02:59@ASH@09
5@MYSTIC@##################################################################
############################################################################
```

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 *Pokémon* deverá ser inserido no final do arquivo de dados e atualizado nos índices.
- **Atualização:** o único campo alterável é o de `combat points`. O registro deverá ser localizado acessando o índice primário e a pontuação deverá ser atualizada no registro na mesma posição em que está (não deve ser feita remoção seguida de inserção). Note que o campo `combat points` sempre terá 7 dígitos.

Índices

Três índices deverão ser criados (um usando árvore-B e dois usando listas) na inicialização do programa e manipulados em RAM até o encerramento da aplicação:

- `iprimary`: índice primário (árvore-B), contendo as chaves primárias e os RRNs dos registros. A ordem será informada pelo usuário e **a promoção deverá ser sempre pelo sucessor imediato** (menor chave da sub-árvore direita).
- `ipokemon`: índice secundário (lista estática ordenada), contendo o nome do *Pokémon* e a chave primária do respectivo registro, ordenado pelo nome do *Pokémon*.
- `iteam`: índice secundário (lista estática ordenada), contendo o nome da equipe do treinador que capturou o *Pokémon* e a chave primária do respectivo registro, ordenado pelo nome da equipe.

Deverá ser desenvolvida uma rotina para a criação de cada índice. Os índices serão sempre criados e manipulados em memória principal na inicialização e liberados ao término do programa. 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 a ordem m da árvore-B e criar os índices na RAM.

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 pergunta pela ordem m da árvore-B usada para indexar as chaves primárias.

As seguintes operações devem ser fornecidas (nessa ordem):

1. **Cadastro.** O usuário deve ser capaz de inserir um novo *Pokémon*. Seu programa deve ler os seguintes campos (nessa ordem): `nome do Pokémon`, `tipo do Pokémon`, `combat points`, `data de captura`, `hora da captura`, `nome do treinador`, `nível do treinador` e `nome da equipe`. Note que a chave **não é** inserida pelo usuário, você precisa gerar a chave para gravá-la no registro. Antes de armazenar um campo, você deve fazer algumas verificações:
   - O `nome do Pokémon` informado deve ser composto apenas por letras.
   - O `tipo do Pokémon` informado deve ser composto apenas por letras e, caso haja mais de um tipo a ser informado, deve ser composto apenas por letras e pelo separador '/' (um tipo não deve iniciar ou terminar com '/').

- Os `combat points` informados devem ser compostos por 7 *bytes* e estar no formato `NNNN.NN`, onde N pertence a [0, 9] e o separador corresponde a '.';
- A `data de captura` informada deve estar no formato "`DD/MM/AA`", onde `AA` é maior ou igual a 16 (referente ao ano corrente e de lançamento do jogo), `MM` pertence ao intervalo [1, 12], `DD` pertence ao intervalo [1, 31], [1, 30], [1, 29] ou [1, 28], de acordo com o mês e o ano, e os dois separadores correspondem a '/';
- A `hora da captura` informada deve estar no formato `HH:MM`, onde `HH` pertence ao intervalo [0, 23], `MM` pertence ao intervalo [0, 59] e o separador corresponde a ':';
- O `nível do treinador` informado deve ser compostos por 3 *bytes* e pertencer ao intervalo [1, 100];
- O `nome da equipe` informado deve sempre corresponder a um dos três nomes a seguir: `Valor`, `Instinct` ou `Mystic` (Como a entrada não tem padrão, recomenda-se que seja feita a conversão das letras para somente maiúsculas ou somente minúsculas antes de fazer a comparação).

Caso algum dos campos esteja irregular, exibir a mensagem "`Campo inválido! Informe novamente: `" e solicitar a digitação novamente. Se a chave primária gerada já existir no arquivo de dados, a seguinte mensagem de erro deverá ser impressa: "`ERRO: Já existe um registro com a chave primária AAAA99999999.\n`", onde `AAAA99999999` corresponde à chave primária do registro que está sendo inserido.

2. **Alteração.** O usuário deve ser capaz de alterar os pontos de combate de um *Pokémon*, informando a sua chave primária. Caso ela não exista, seu programa deverá exibir a mensagem "`Registro não encontrado!`" e retornar ao menu. Caso o registro seja encontrado, certifique-se de que o novo valor informado está dentro dos padrões (7 *bytes*, no formato `NNNN.NN`, com `N` pertencente ao intervalo [1, 9] e um ponto '.' como separador) e, nesse caso, altere o valor do campo no arquivo de dados. Caso contrário, exiba a mensagem "`Campo inválido!`" e solicite a digitação novamente.

3. **Busca.** O usuário deve ser capaz de buscar por uma partida:

   - **1. por código:** solicitar ao usuário a chave primária. Caso o *Pokémon* não exista, seu programa deve exibir a mensagem "`Registro nao encontrado!`" e retornar ao menu principal. Caso o *Pokémon* exista, todos os dados do *Pokémon* devem 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 árvore-B de ordem $m = 3$ composta pelos seguintes registros inseridos na ordem `IBGE10081309`, `IBON10101437`, `ICPI19051857`, `MABU13081125`, `MACH30042359`, `MAPI08080500`, `VMPS26072211`, `VMTO29021456`, `VPBU11070943` e `VPCH31080000` (Figura 1), a busca pela chave `IBGE10081309` e `MABU99999999` retornará:

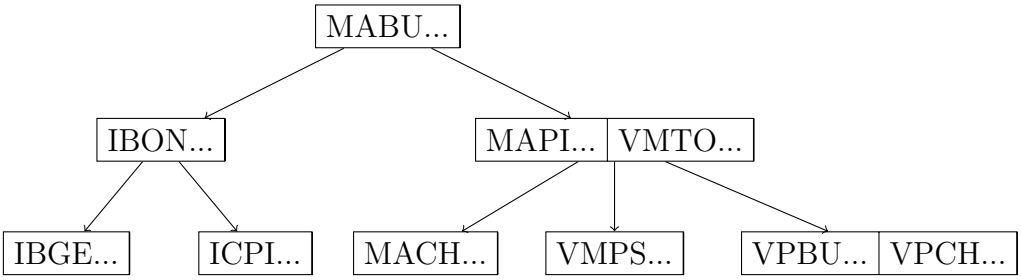

Figura 1: Índice primário estruturado em Árvore-B de ordem 3.

```
Busca por IBGE10081309. Nos percorridos:
MABU13081125
IBON10101437
IBGE10081309

IBGE10081309
GEODUDE
PEDRA/TERRA
0808.00
10/08/2016
13:09
BROCK
088
INSTINCT

Busca por MABU99999999. Nos percorridos:
MABU13081125
MAPI08080500, VMTO29021456
MACH30042359

Registro nao encontrado!
```

- **2. por nome de *Pokémon*:** solicitar ao usuário o nome do *Pokémon*. Caso o *Pokémon* informado não tiver sido capturado por nenhum jogador, o programa deve exibir a mensagem "`Registro nao encontrado!`" e retornar ao menu principal. Caso existam um ou mais *Pokémons* cujo nome seja o informado pelo usuário, os registros completos desses *Pokémons* deverão ser mostrados na tela de forma formatada, ordenados pela chave primária e separados por uma linha vazia.
- **3. por nome da equipe:** solicitar ao usuário o nome da equipe. Caso a equipe não tenha capturado nenhum *Pokémon*, o programa deve exibir a mensagem "`Registro não encontrado!`" e retornar ao menu principal. Caso existam um ou mais *Pokémons* que pertençam à equipe informada, os registros completos desses *Pokémons* deverão ser mostrados na tela de forma formatada, ordenados pela chave primária e separados por uma linha vazia.

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

   - **1. árvore-B:** imprime a árvore-B (somente o campo de chave primária) usando varredura **pré-ordem**. Imprimir um nó por linha, começando pelo nível da árvore em que se encontra o nó (a partir da raiz – nível 1) seguido da chave. **Na última linha, adicione uma quebra de linha adicional.** Por exemplo, considere a árvore-B apresentada na Figura 1, a sua listagem resultaria em:

     1 – `MABU13081125`
     2 – `IBON10101437`
     3 – `IBGE10081309`
     3 – `ICPI19051857`
     2 – `MAPI08080500`, `VMTO29021456`
     3 – `MACH30042359`
     3 – `VMPS26072211`
     3 – `VPBU11070943`, `VPCH31080000`

   - **2. por nome do *Pokémon*:** exibe os *Pokémons* na ordem lexicográfica de seus nomes.
   - **3. por nome de equipe:** exibe os *Pokémons* na ordem lexicográfica de nome da equipe.

   As listagens 2 e 3 deverão exibir todos os registros de maneira formatada separados por uma linha vazia. Caso o arquivo de dados esteja vazio, o programa deve exibir a mensagem "`Arquivo vazio!`" e retornar ao menu.

5. **Finalizar.** Encerra a execução do programa. Ao final da execução, feche o arquivo de dados e libere toda a memória alocada pelo programa.

Implementação

Implemente suas funções utilizando como base o código fornecido. Não modifique os trechos de código ou as estruturas já prontas. 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 o caso de teste.

Você deve criar obrigatoriamente as seguintes funcionalidades:

- Criar o índice primário (árvore-B): deve construir o índice primário a partir do arquivo de dados e da ordem m informada na inicialização do programa;
- Criar os índices secundários (índices simples): deve construir os índices secundários a partir do arquivo de dados;
- Inserir um registro: modificar o arquivo de dados e os índices na memória principal.
- Buscar por registros: buscar por registros pela chave primária ou por uma das chaves secundárias.
- Alterar um registro: modificar o arquivo de dados.
- Listar registros: listar a árvore-B ou todos os registros ordenados por uma das chaves secundárias.
- Finalizar: deverá ser chamada ao encerrar o programa e liberar toda a memória alocada.

Utilizar a linguagem `ANSI C`.

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.