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

TRABALHO 01 - INDEXAÇÃO

Atenção

1. **Prazo de entrega: 18/10/2017 às 23h55 (via OnlineJudge – AVA)**
2. A atividade deverá ser realizada individualmente

Organização de arquivos e indexação

Steam é considerada a maior plataforma de distribuição de jogos digitais. Ela contém cerca de 65 milhões de usuários e mais de 17.000 títulos de jogos. Todo ano ocorrem as famosas temporadas de vendas, na qual são apresentados diversos títulos em promoção. Devido ao alto número de usuários interessados em obter novos títulos, o sistema acaba ficando super congestionado. Consequentemente, ela precisa dos seus conhecimentos para otimizar a pesquisa por seus títulos, utilizando técnicas de indexação. Para cada título é preciso armazenar os seguintes dados:

- *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|INDIE|CORRIDA|ESPORTES`);

Garantidamente, nenhum campo de texto receberá caractere acentuado.

Tarefa

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

1. Inserir um novo título;
2. Remover um título a partir da chave primária;
3. Modificar o campo **Desconto** de um título a partir da chave primária;
4. Buscar jogos a partir:
   1) da chave primária;
   2) do título do jogo; ou
   3) da categoria e desenvolvedora.
5. Listar todos os jogos ordenados por:
   1) código;
   2) categoria e chave primária;
   3) desenvolvedora e chave primária;
   4) por preço com desconto aplicado (em ordem crescente) e chave primária.
6. Liberar espaço.
7. Imprimir arquivo de dados.
8. Finalizar (liberar estruturas da memória RAM).

Para realizar essa tarefa será necessário organizar e manter pelo menos um arquivo e cinco índices distintos:

(a) um arquivo de dados que conterá todos os registros;

(b) um índice primário;

(c) um índice secundário para os títulos dos jogos;

(d) um índice secundário para as categorias;

(e) um índice secundário para as desenvolvedoras; e

(f) um índice secundário para preços com descontos aplicados.

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 *strings* que simularão os arquivos abertos. 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.

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.

```
LIFE IS STRANGE: BEFORE THE STORM@DECK NINE@31/08/2017@18@0044.9
9@008@ACAO|AVENTURA@##################################################
############################################################AB
SOLVER@SLOCLAP@29/08/2017@13@0055.99@000@ACAO|AVENTURA|INDIE@###
############################################################
############################################################STREE
T FIGHTER V@CAPCOM@16/02/2016@14@0089.99@010@ACAO@##############
############################################################
##########################################################THE WITC
HER 3: WILD HUNT - GOTY EDITION@CD PROJEKT RED@30/08/2016@18@009
9.99@050@RPG@#############################################################
#####################################################GRAND THEFT
 AUTO V@ROCKSTAR@14/04/2015@18@0299.99@100@ACAO|AVENTURA@#######
############################################################
##################################################
```

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.
- **Remoção:** o registro deverá ser localizado acessando o índice primário. A remoção deverá colocar os caracteres `*|` nas primeiras posições do registro removido. O espaço do registro removido não deverá ser reutilizado para novas inserções. Observe que o registro deverá continuar ocupando exatamente 192 bytes. Além disso, no índice primário, o RRN correspondente ao registro removido deverá ser substituído por -1.
- **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.

Índices

Pelo menos índices deverão ser criados:

- `iprimary`: índice primário, contendo a chave primária e o RRN do respectivo registro, ordenado pela chave primária;
- `igame`: índice secundário, contendo o nome do jogo e a chave primária do respectivo registro, ordenado pelo nome do jogo e em caso de empate, pelo código.
- `idev`: índice secundário, contendo a desenvolvedora e a chave primária do respectivo registro, ordenado pelo nome da desenvolvedora. Em caso de jogos desenvolvidos pela mesma desenvolvedora, ordenar pelo código.
- `icat`: índice secundário, contendo a categoria do jogo e a chave primária do respectivo registro, ordenado pelo nomes da categorias em seguida pelo código.
- `iprice`: índice secundário, contendo o preço final e o código do jogo, deve ser ordenado primeiramente pelo preço em ordem ascendente e, em seguida, pelo código.

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 `iprimary` seja o primeiro a ser criado. Quanto aos índices secundários, fica de livre escolha optar por usar índice simples ou invertido.

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.

Interação com o usuário

O programa deve permitir interação com o usuário pelo console/terminal (modo texto) via menu. 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 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 (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. **Remoção.** O usuário deve ser capaz de remover um jogo. Caso ele não exista, seu programa deverá exibir a mensagem "`Registro não encontrado!\n`" e retornar ao menu. Para remover um jogo, seu programa deverá solicitar como entrada ao usuário somente o campo chave primária e a remoção deverá ser feita por um marcador.

4. **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 não encontrado!\n`" e retornar ao menu principal. Caso o jogo exista, todos os seus dados devem ser impressos na tela de forma formatada, exibindo os campos na mesma ordem de inserção.
    - **2. por título do jogo:** solicitar ao usuário o nome completo do jogo. Caso o jogo não tenha sido encontrado, o programa deve exibir a mensagem "`Registro não encontrado!\n`" e retornar ao menu principal. Caso existam um ou mais jogos que possuem o mesmo nome, os registros completos desses deverão ser mostrados na tela de forma formatada, ordenados pela chave primária e separados por uma linha vazia.
    - **3. por nome da desenvolvedora e categoria:** solicitar ao usuário o nome da desenvolvedora e logo em seguida a categoria. Caso a desenvolvedora informada não tenha nenhum jogo cadastrado na categoria em questão, o programa deve exibir a mensagem "`Registro(s) não encontrado!\n`" e retornar ao menu principal. Caso existam um ou mais jogos que estão nos critérios informados pelo usuário, os registros completos desses deverão ser mostrados na tela de forma formatada, ordenados pela chave primária e separados por uma linha vazia.

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

    - **1. por código:** exibe os jogos na ordem lexicográfica de código.
    - **2. por categoria:** exibe todos os jogos da categoria informada na ordem lexicográfica do código.
    - **3. por desenvolvedora:** exibe os jogos de todas as desenvolvedoras na ordem lexicográfica do da desenvolvedora, seguida pela do código.
    - **4. por preço com desconto aplicado:** exibe os jogos em ordem crescente de valor final. Em caso de empate, apresentar em ordem lexicográfica do código. Nessa listagem, e somente nesta listagem, não mostre o preço inicial e o desconto, apresente apenas o preço final.

As listagens deverão exibir todos os registros (exceto os marcados como excluídos) de maneira formatada e separados por uma linha vazia. Caso nenhum registro tenha sido apresentado, o programa deve exibir a mensagem "`Registro(s) não encontrado!\n`" e retornar ao menu.

6. **Liberar espaço.** O arquivo de dados deverá ser reorganizado com a remoção física de todos os registros marcados como excluídos e os índices deverão ser atualizados. A ordem os registros no arquivo limpo não deverá ser diferente da original.

7. **Imprimir arquivo de dados.** Imprime o arquivo de dados. Caso esteja vazio, apresente a mensagem "`Arquivo Vazio!`"

8. **Finalizar.** Liberar memória e encerra a execução do programa.

Implementação

Implementar uma biblioteca de manipulação de arquivos para o seu programa, contendo obrigatoriamente as seguintes funcionalidades:

- Estrutura de dados adequadas para armazenar os índices na memória principal;
- Verificar se o arquivo de dados existe;
- Criar o índice primário: deve refazer o índice primário a partir do arquivo de dados;
- Criar os índices secundários: deve refazer os índices secundários a partir do arquivo de dados;
- Inserir um registro: modificando o arquivo de dados e os índices na memória principal.
- Buscar por registros: busca pela chave primária ou por uma das chaves secundárias.
- Alterar um registro: modificando o campo do registro diretamente no arquivo de dados.
- Remover um registro: modificando o arquivo de dados e o índice primário na memória principal.
- Listar registros: listar todos os registros ordenados pela chave primária ou por uma das chaves secundárias.
- Liberar espaço: organizar o arquivo de dados e refazer os índices.

Dicas

- Ao ler entrada, tome cuidado com caracteres de quebra de linha (\n) não capturados.
- 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 manualmente. Você também pode utilizar o comando `strncpy` para escrever em strings, esse comando, diferentemente do `sprintf`, não adiciona o caractere nulo no final.
- A função `strtok` permite navegar nas substrings de uma certa string dado 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.
- É permitido (e recomendado) utilizar as funções de `qsort` e `bsearch`. Leia atentamente a documentação antes de usa-las.
- Para o funcionamento ideal do seu programa, é necessário utilizar a busca binária, porém tenha em mente que ela não garante que irá retornar o primeiro elemento do tipo, caso haja repetição.
- Caso a sua submissão esteja retornando funções restringidas é possível que a sua implementação possua funções que estejam acessando memória indevida.

CUIDADOS

Leia atentamente os itens a seguir.

O projeto deverá ser enviado pelo ambiente virtual de aprendizagem observando as seguintes regras:

1. Submeter um único arquivo `.c` chamado `{RA}_ED2_T01.c`, ex: `123456_ED2_T01.c`
2. Utilizar a linguagem `ANSI C`;
3. **Identificadores de variáveis:** escolha nomes apropriados;
4. **Documentação:** inclua cabeçalho, comentários e indentação no programa;
5. **Dificuldades em implementação:** consultar o monitor da disciplina nos horários estabelecidos;
6. **Erros de compilação:** nota **zero** no trabalho;
7. **Tentativa de fraude:** nota **zero na média** para todos envolvidos. Enfatizo que a detecção de cópia de parte ou de todo código-fonte, de qualquer origem, implicará na reprovação direta na disciplina. Partes do código cujas **ideias** foram desenvolvidas em colaboração com outro(s) aluno(s) devem ser devidamente documentadas em comentários no referido trecho. O que **NÃO** autoriza a cópia de trechos de código nem a codificação em conjunto. Portanto, compartilhem ideias, soluções, modos de resolver o problema, mas não o código.