Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
Disciplina de Algoritmos e Estruturas de Dados II
docente
Profa. Dra. Cristina Dutra de Aguiar Ciferri (cdac@icmc.usp.br)
aluno PAE
Victor Hugo Andrade Soares (victorhugoasoares@usp.br)
monitor
João Vitor dos Santos Tristão (joaovitortristao@usp.br)

Quinto Trabalho Prático

Este trabalho tem como objetivo indexar arquivos de dados usando um índice árvore-B.

*O trabalho deve ser feito em **dupla** ou **individualmente**. A solução deve ser proposta exclusivamente pelo(s) aluno(s) com base nos conhecimentos adquiridos nas aulas. Consulte as notas de aula e o livro texto quando necessário. O(s) aluno(s) deve(m) escolher um dos trabalhos anteriores a ser utilizado para desenvolver o trabalho prático 5. Isso deve ser informado no início do código do trabalho prático 5 como um comentário.*

Descrição de páginas de disco

No trabalho será usado o conceito de páginas de disco. Cada página de disco tem o tamanho fixo de 93 bytes. O conceito de página de disco é um conceito lógico, ou seja, deve ser garantido via programação, de forma que cada página de disco contenha, no máximo, o tamanho fixo especificado.

Importante. Na prática, o tamanho de página de disco do arquivo de dados e do arquivo de índice da árvore-B é o mesmo. Entretanto, como foi definido um tamanho de página de disco muito grande nos trabalhos práticos anteriores (ou seja, de 16.000 bytes), optou-se por se usar dois tamanhos de páginas de disco diferentes: de 16.000 bytes para o arquivo de dados e de 93 bytes para o arquivo de índice da árvore-B. Dessa forma, os arquivos já implementados até então podem ser utilizados sem necessidade de alteração.

Descrição do arquivo de índice árvore-B

O índice árvore-B com ordem *m* é definido formalmente como descrito a seguir.

1. Cada página (ou nó) do índice árvore-B deve ser, pelo menos, da seguinte forma:

   $<P_1, <C_1,P_{R1}>, P_2, <C_2,P_{R2}>, ..., P_{q-1}, <C_{q-1},P_{Rq-1}>, P_q>$, onde $(q \leq m)$ e

   - Cada P_j $(1 \leq j \leq q)$ é um ponteiro para uma subárvore ou assume o valor -1 caso não exista subárvore (ou seja, caso seja um nó folha).
   - Cada C_i $(1 \leq i \leq q - 1)$ é uma chave de busca.
   - Cada P_{Ri} $(1 \leq i \leq q - 1)$ é um campo de referência para o registro no arquivo de dados que contém o registro de dados correspondente a C_i.
2. Dentro de cada página (ou seja, as chaves de busca são ordenadas)
   - $C_1 < C_2 < ... < C_{q-1}$.
3. Para todos os valores X da chave na subárvore apontada por P_i:
   - $C_{i-1} < X < C_i$ para $1 < i < q$
   - $X < K_i$ para $i = 1$
   - $K_{i-1} < X$ para i = q.
4. Cada página possui um máximo de *m* descendentes.
5. Cada página, exceto a raiz e as folhas, possui no mínimo $\lceil m/2 \rceil$ descendentes (*taxa de ocupação*).
6. A raiz possui pelo menos 2 descendentes, a menos que seja um nó folha.
7. Todas as folhas aparecem no mesmo nível.
8. Uma página não folha com *k* descendentes possui *k*-1 chaves.
9. Uma página folha possui no mínimo $\lceil m/2 \rceil - 1$ chaves e no máximo $m - 1$ chaves (*taxa de ocupação*).

Descrição do Registro de Cabeçalho. O registro de cabeçalho deve conter os seguintes campos:

- *status*: indica a consistência do arquivo de índice, devido à queda de energia, travamento do programa, etc. Pode assumir os valores 0, para indicar que o arquivo de dados está inconsistente, ou 1, para indicar que o arquivo de dados está consistente. Ao se abrir um arquivo para escrita, seu status deve ser 0 e, ao finalizar o uso desse arquivo, seu status deve ser 1 – tamanho: *string* de 1 byte.
- *noRaiz*: armazena o RRN do nó (página) raiz do índice árvore-B – tamanho: inteiro de 4 *bytes*

Representação Gráfica do Registro de Cabeçalho. O tamanho do registro de cabeçalho deve ser de 5 bytes, representado da seguinte forma:

1 byte	4 bytes
status	*noRaiz*

Página de disco. O registro de cabeçalho deve ocupar uma página de disco. Seu tamanho é menor do que o tamanho da página de disco. Neste caso, a página de disco deve ser preenchida com caractere '@' até completar o seu tamanho.

Descrição do Registro de Dados. Deve ser considerada a seguinte organização: campos de tamanho fixo e registros de tamanho fixo. Em adição ao Item 1 da definição formal do índice árvore-B, cada nó (página) da árvore também deve armazenar dois outros campos:

- *eFolha*, indicando se o nó é um nó folha ou não, representado por uma *string* de 1 byte (0 indica que o nó não é folha e 1 indica que o nó é folha); e
- *n*, indicando o número de chaves do nó, representado por um inteiro de 4 bytes.

A ordem da árvore-B é 8, ou seja, $m = 8$. Portanto, um nó (página) terá 7 chaves e 8 ponteiros. A chave de busca é o campo *nroInscricao*.

Representação Gráfica de um Nó (Página/Registro de Dados) do índice. O tamanho de cada registro de dados é de 93 bytes, representado da seguinte forma:

1 byte	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	...	4 bytes	4 bytes	4 bytes
eFolha	*n*	P_1	C_1	P_{R1}	P_2	C_2	P_{R2}	...	C_7	P_{R7}	P_8
0	1	...									92

Observações Importantes.

- Cada registro de dados deve seguir estritamente a ordem definida na sua representação gráfica.
- Os nomes dos atributos também devem seguir estritamente os nomes definidos na especificação dos mesmos.
- Quando um nó (página) do índice tiver chaves de busca que não forem preenchidas, deve ser armazenado o caractere '@'. O valor -1 deve ser usado para denotar que um ponteiro *P* de um nó é nulo.

Programa

Descrição Geral. Implemente um programa em C que ofereça uma interface por meio da qual o usuário possa realizar operações sobre um índice árvore-B e que possa inserir e buscar dados de um arquivo de dados que está indexado por esse índice simples ou linear.

Importante. A definição da sintaxe de cada comando bem como sua saída devem seguir estritamente as especificações definidas em cada funcionalidade. Para especificar a sintaxe de execução, considere que o programa seja chamado de "programaTrab5". Essas orientações devem ser seguidas uma vez que a correção do funcionamento do programa se dará de forma automática.

Descrição Específica. O programa deve oferecer as seguintes funcionalidades:

[16] Crie um arquivo de índice árvore-B para um arquivo de dados de entrada já existente. O campo a ser indexado é *nroInscricao*. Registros logicamente removidos presentes no arquivo de dados de entrada não devem ter suas chaves de busca correspondentes no arquivo de índice. A inserção no arquivo de índice deve ser feita um-a-um. Ou seja, para cada registro não removido presente no arquivo de dados, deve ser feita a inserção de sua chave de busca correspondente no arquivo de índice árvore-B. A manipulação do arquivo de índice árvore-B deve ser feita em disco, de acordo com o conteúdo ministrado em sala de aula. Antes de terminar a execução da funcionalidade, deve ser utilizada a função binarioNaTela1 ou binarioNaTela2, ambas disponibilizadas na página do projeto da disciplina, para mostrar a saída do arquivo de índice árvore-B.

Entrada do programa para a funcionalidade [15]:

```
16 arquivoEntrada.bin arquivoIndiceNroInscricao.bin
```

onde:

- `arquivoEntrada.bin` é um arquivo binário de entrada que segue as mesmas especificações do primeiro trabalho prático, e que contém dados desordenados e registros logicamente removidos.
- `arquivoIndiceNroInscricao.bin` é o arquivo binário de índice árvore-B que indexa o campo *nroInscricao*. Esse arquivo deve seguir as especificações definidas neste trabalho prático.

Saída caso o programa seja executado com sucesso:

```
Listar o arquivo binário arquivoIndiceNroInscricao.bin.
```

Mensagem de saída caso algum erro seja encontrado:

```
Falha no processamento do arquivo.
```

Exemplo de execução:

```
./programaTrab5
16 arquivoEntrada.bin arquivoIndiceNroInscricao.bin
usar a função binarioNaTela1 ou binarioNaTela2 antes de terminar a execução da funcionalidade, para mostrar a saída do arquivo arquivoIndiceNroInscricao.bin, o qual indexa o campo nroInscricao.
```

[17] Permita a recuperação dos dados de todos os registros que satisfaçam um critério de busca determinado pelo usuário sobre o campo *nroInscricao*, usando o índice árvore-B criado na funcionalidade [16]. Note que somente o campo *nroInscricao* deve ser utilizado como forma de busca, desde que o índice criado na funcionalidade [16] indexa chaves de busca desse campo. Essa funcionalidade pode retornar 0 registros (quando nenhum satisfaz ao critério de busca) ou 1 registro (desde que o identificador do servidor não aceita valores repetidos). A manipulação do arquivo de índice árvore-B deve ser feita em disco, de acordo com o conteúdo ministrado em sala de aula. Depois de mostrar todos os registros, deve ser mostrado na saída padrão o número de níveis do arquivo árvore-B que foram percorridos. Por exemplo, a busca começa pelo nó raiz (número de níveis = 1); segue pela subárvore mais à esquerda (número de níveis = 2); e segue pela subárvore mais à direita (número de níveis = 3).

Sintaxe do comando para a funcionalidade [16]:

```
17 arquivo.bin arquivoIndiceNroInscricao.bin nroInscricao valor
```

Saída caso o programa seja executado com sucesso:

O registro deve ser mostrado em uma única linha e os seus campos devem ser mostrados de forma sequencial separado por espaço. Campos de tamanho fixo que tiverem o valor nulo não devem mostrados. Para os campos com tamanho variável, mostre também sua quantidade de caracteres (tamanhos em bytes sem o caractere '\0'). Para os campos de tamanho variável com valores nulos, não deve ser exibido nada. Especifique também o número de níveis do arquivo árvore-B que foram percorridos.

Mensagem de saída caso não seja encontrado o registro que contém o valor do campo ou o campo pertence a um registro que esteja removido:

```
Registro inexistente.
```

Mensagem de saída caso algum erro seja encontrado:

```
Falha no processamento do arquivo.
```

Exemplo de execução (considere que os metadados lidos do arquivo .csv sejam: numero de identificacao do servidor, salario do servidor, telefone celular do servidor, nome do servidor, cargo do servidor):

```
./programaTrab5
17 arquivo.bin arquivoIndiceNroInscricao.bin nroInscricao 2104
2104 457 13 BARROS CONEGO
Número de níveis do índice árvore-B percorridos: X
```

Restrições

As seguintes restrições têm que ser garantidas no desenvolvimento do trabalho.

[1] O arquivo de dados deve ser gravado em disco no **modo binário**. O modo texto não pode ser usado.

[2] Os dados do registro descrevem os nomes dos campos, os quais não podem ser alterados. Ademais, todos os campos devem estar presentes na implementação, e nenhum campo adicional pode ser incluído. O tamanho e a ordem de cada campo deve obrigatoriamente seguir a especificação.

[3] Deve haver a manipulação de valores nulos, conforme as instruções definidas.

[4] Não é necessário realizar o tratamento de truncamento de dados.

[5] Devem ser exibidos avisos ou mensagens de erro de acordo com a especificação de cada funcionalidade.

[6] Os dados devem ser obrigatoriamente escritos e lidos campo a campo. Ou seja, não é possível escrever e ler os dados registro a registro.

[7] O(s) aluno(s) que desenvolveu(desenvolveram) o trabalho prático deve(m) constar como comentário no início do código (i.e. NUSP e nome do aluno). Para trabalhos desenvolvidos por mais do que um aluno, não será atribuída nota ao aluno cujos dados não constarem no código fonte.

[8] Todo código fonte deve ser documentado. A **documentação interna** inclui, dentre outros, a documentação de procedimentos, de funções, de variáveis, de partes do código fonte que realizam tarefas específicas. O código fonte deve ser documentado tanto em nível de rotinas quanto em nível de variáveis e blocos funcionais.

[9] A implementação deve ser realizada usando a linguagem de programação C. As funções das bibliotecas <stdio.h> devem ser utilizadas para operações relacionadas à escrita e leitura dos arquivos. A implementação não pode ser feita em qualquer outra linguagem de programação. O programa executará no [run.codes].

Fundamentação Teórica

Conceitos e características dos diversos métodos para representar os conceitos de campo e de registro em um arquivo de dados podem ser encontrados nos *slides* de sala de aula e também nas páginas 96 a 107 do livro *File Structures* (*second edition*), de Michael J. Folk e Bill Zoellick.

Material para Entregar

Arquivo compactado. Deve ser preparado um arquivo .zip contendo:

- Código fonte do programa devidamente documentado.
- Makefile para a compilação do programa.

Instruções para fazer o arquivo makefile. No [run.codes] tem uma orientação para que, no makefile, a diretiva "all" contenha apenas o comando para compilar seu programa e, na diretiva "run", apenas o comando para executá-lo. Assim, a forma mais simples de se fazer o arquivo makefile é:

```
all:
gcc -o programaTrab5 *.c
run:
./programaTrab5
```

Lembrando que *.c já engloba todos os arquivos .c presentes no seu zip."

Instruções de entrega. A entrega deve ser feita via [run.codes]:

- página: https://run.codes/Users/login
- código de matrícula: **91X6**

Critério de Correção

Critério de avaliação do trabalho. Na correção do trabalho, serão ponderados os seguintes aspectos.

- Corretude da execução do programa.
- Atendimento às especificações do registro de cabeçalho e dos registros de dados.
- Atendimento às especificações da sintaxe dos comandos de cada funcionalidade e do formato de saída da execução de cada funcionalidade.
- Qualidade da documentação entregue.

Restrições adicionais sobre o critério de correção.

- A não execução de um programa devido a erros de compilação implica que a nota final da parte do trabalho será igual a zero (0).
- O não atendimento às especificações do registro de cabeçalho e dos registros de dados implica que haverá uma diminuição expressiva na nota do trabalho.
- O não atendimento às especificações de sintaxe dos comandos de cada funcionalidade e do formato de saída da execução de cada funcionalidade implica que haverá uma diminuição expressiva na nota do trabalho.
- A ausência da documentação implica que haverá uma diminuição expressiva na nota do trabalho.
- A inserção de palavras ofensivas nos arquivos e em qualquer outro material entregue implica que a nota final da parte do trabalho será igual a zero (0).
- Em caso de plágio, as notas dos trabalhos envolvidos serão zero (0).

Data de Entrega do Trabalho

Na data especificada na página da disciplina.

Bom Trabalho !