Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
Disciplina de Estrutura de Dados III (SCC0607)

docente
Profa. Dra. Cristina Dutra de Aguiar Ciferri (cdac@icmc.usp.br)
alunos PAE
João Paulo Clarindo (jpcsantos@usp.br)
monitores
Helbert Moreira Pinto [helbert.moreira@usp.br] telegram: @HelbertMP
Matheus Carvalho Raimundo [mcarvalhor@usp.br] telegram: @mcarvalhor

Segundo Trabalho Prático

Este trabalho tem como objetivo armazenar dados em arquivos binários de acordo com uma organização de campos e registros, bem como recuperar os dados armazenados com e sem o uso de um índice primário.

De acordo com o critério de avaliação da disciplina, o trabalho deve ser feito por, no máximo, ***2 alunos, que são os mesmos alunos do trabalho prático 1. Qualquer mudança deve ser informada e conversada com a docente ou com os monitores****. 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.*

Descrição dos arquivos de dados
* Fundamentos da disciplina de Bases de Dados *

O trabalho tem como objetivo armazenar e recuperar dados relacionados a pessoas que seguem pessoas no **twitter**. Nesse sentido, a disciplina de Estrutura de Dados III é uma disciplina fundamental para a disciplina de Bases de Dados. Na disciplina de Bases de Dados, é ensinado que o projeto deve ser feito em dois arquivos. O primeiro arquivo é relacionado às pessoas, armazenando apenas dados relacionados a essas pessoas. O segundo arquivo é um arquivo de relacionamento, que relaciona pessoas que seguem outras pessoas. Visando atender aos requisitos de um bom projeto do banco de dados, são definidos dois arquivos de dados a serem utilizados nos trabalhos práticos: arquivo de dados **pessoa** e arquivo de dados **segue**. Neste segundo trabalho prático, é implementado o arquivo **segue**.

Descrição do arquivo de dados segue

O arquivo de dados **segue** possui um registro de cabeçalho e 0 ou mais registros de dados, conforme a definição a seguir.

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

- *status*: indica a consistência do arquivo de dados, 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.
- *quantidadeSeguidores*: armazena a quantidade de pessoas que seguem outras pessoas (ou o número de registros) presentes no arquivo – tamanho: inteiro de 4 bytes.

Representação Gráfica do Registro de Cabeçalho. O tamanho do registro de cabeçalho deve ser de 32 bytes, representado da seguinte forma:

<table>
<tr><td>1 byte</td><td colspan="4">4 bytes</td><td colspan="4">27 bytes</td></tr>
<tr><td>status</td><td colspan="4">quantidade Seguidores</td><td colspan="4">lixo (caractere '$')</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>...</td><td>29</td><td>30</td><td>31</td></tr>
</table>

Observações Importantes.

- O registro de cabeçalho deve seguir estritamente a ordem definida na sua representação gráfica.
- Para caber em páginas de disco (que são definidas em potência de 2), o registro de cabeçalho foi definido como uma potência de 2, no mesmo tamanho dos registros de dados). Portanto, o registro de cabeçalho tem o tamanho de 32 bytes, sendo 27 bytes preenchidos com dados necessários para o desenvolvimento do trabalho, e os bytes restantes preenchidos com lixo. O lixo é representado pelo caractere '$'.

Registros de Dados. Os registros de dados são de tamanho fixo, com campos de tamanho fixo, da seguinte forma:

- *removido*: indica se o registro está logicamente removido. Pode assumir os valores '0', para indicar que o registro está marcado como removido, ou '1', para indicar que o registro não está marcado como removido. Ao se inserir um novo registro, o valor de *removido* deve ser '1'– tamanho: *string* de 1 byte.
- *idPessoaQueSegue*: código que identifica univocamente cada pessoa que segue *outra pessoa – inteiro – tamanho: 4 bytes.*
- *idPessoaQueESeguida: código* que identifica univocamente cada pessoa que é seguida por outra pessoa – inteiro – tamanho: 4 bytes.
- *grauAmizade*: indica o grau de amizade que a pessoa que segue tem pela pessoa que é seguida. Pode assumir os valores '0' (segue porque é uma celebridade), '1' (segue porque é amiga de minha amiga), '2' (segue porque é minha amiga) – tamanho: *string* de 3 bytes. Portanto, deve ser armazenado '0\0$', '1\0$', '2\0$' ou '\0$$'.
- *dataInicioQueSegue*: data de início que a pessoa *idPessoaQueSegue* começou a seguir a pessoa *idPessoaQueESeguida – string* – tamanho: 10 bytes, no formato DD/MM/AAAA.
- *dataFimQueSegue*: data na qual a pessoa identificada por *idPessoaQueSegue* parou de seguir a pessoa *idPessoaQueESeguida – string* - tamanho: 10 bytes, no formato DD/MM/AAAA.

Os dados são fornecidos juntamente com a especificação deste trabalho prático por meio de um arquivo .csv, sendo que sua especificação se encontra disponível na página da disciplina. No arquivo .csv, o separador de campos é vírgula (,).

Representação Gráfica dos Registros de Dados. O tamanho de cada registro de dados deve ser de 32 bytes, representado da seguinte forma:

1 byte	4 bytes	4 bytes	3 bytes	10 bytes	10 bytes
removido	*idPessoa QueSegue*	*idPessoa QueESeguida*	*grauAmizade*	*dataInicio QueSegue*	*dataFim QueSegue*

0	1	2	3	4	5	6	7	8	9	10	11	12	...	29	30	31

Observações Importantes.

- Cada registro de dados deve seguir estritamente a ordem definida na sua representação gráfica.
- Deve ser feita a diferenciação entre o espaço utilizado e o lixo. Quando necessário, as *strings* devem ser finalizadas com '\0' e o lixo deve ser identificado pelo caractere '$'. Nenhum *byte* do registro deve permanecer vazio, ou seja, cada *byte* deve armazenar um valor válido ou '\0' ou '$'.
- Nenhum campo aceita valores nulos. O arquivo .csv com os dados de entrada já garante essa característica.
- Não é necessário realizar o tratamento de truncamento de dados.
- Para caber em páginas de disco (que são definidas em potência de 2), os registros de dados foram definidos como uma potência de 2. Ou seja, cada registro de dados tem tamanho de 32 bytes.

Programa

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 "programaTrab". Essas orientações devem ser seguidas uma vez que a correção do funcionamento do programa se dará de forma automática. De forma geral, a primeira entrada da entrada padrão é sempre o identificador de suas funcionalidades, conforme especificado a seguir.

Descrição Geral. Implemente um programa em C que ofereça as seguintes funcionalidades.

[6] Permita a leitura de vários registros obtidos a partir de um arquivo de entrada (arquivo no formato .csv) e a gravação desses registros em um arquivo de dados de saída. O arquivo de entrada é fornecido juntamente com a especificação do projeto, enquanto o arquivo de dados de saída (arquivo **segue**) deve ser gerado como parte deste trabalho prático. Nessa funcionalidade, ocorrem várias inserções no arquivo **segue**. Antes de terminar a execução da funcionalidade, deve ser utilizada a função binarioNaTela, disponibilizada na página do projeto da disciplina, para mostrar a saída do arquivo binário. Lembre-se de manipular o campo "status" do registro de cabeçalho adequadamente. Primeiro ele deve começar com o valor '0' (arquivo inconsistente) e, só ao final da execução do programa e ao final de todas as gravações, ele deve mudar para o valor '1' (arquivo consistente).

Entrada do programa para a funcionalidade [6]:

```
6 arquivoEntrada.csv arquivoSegue.bin
```

onde:

- `arquivoEntrada.csv` é um arquivo .csv que contém os valores dos campos dos registros a serem armazenados no arquivo de dados **segue**.
- `arquivoSegue.bin` é o arquivo de dados **segue** no formato binário gerado conforme as especificações descritas neste trabalho prático.

Saída caso o programa seja executado com sucesso:

```
Listar o arquivo de dados no formato binário usando a função fornecida binarioNaTela.
```

Mensagem de saída caso algum erro seja encontrado:

```
Falha no carregamento do arquivo.
```

Exemplo de execução:

```
./programaTrab
6 arquivoEntrada.csv arquivoSegue.bin
usar a função binarioNaTela antes de terminar a execução da funcionalidade, para mostrar a saída do arquivo binário arquivoSegue.bin
```

[7] Ordene o arquivo de dados de acordo com o campo de ordenação *idPessoaQueSegue*. Quando houver empate para os valores de *idPessoaQueSegue*, deve ser usado como critério de desempate o campo *idPessoaQueESeguida* e, caso ainda haja empate, deve ser usado como critério de desempate o campo *dataInicioQueSegue* e, caso ainda haja empate, deve ser usado como critério de desempate o campo *dataFimQueSegue*. A ordenação deve ser implementada considerando que o arquivo de dados **segue** cabe totalmente em memória primária (RAM). Portanto, o arquivo deve ser: (i) lido inteiramente do disco para a RAM; (ii) ordenado de acordo com a chave de ordenação usando-se qualquer algoritmo de ordenação disponível na biblioteca da linguagem C; e (iii) escrito inteiramente para disco novamente, gerando um novo arquivo de dados **segueOrdenado**, que encontra-se ordenado de acordo com o campo de ordenação. O arquivo de dados **segue** original não deve ser removido.

Entrada do programa para a funcionalidade [7]:

7 arquivoSegue.bin arquivoSegueOrdenado.bin

onde:

- arquivoSegue.bin é o arquivo de dados **segue** no formato binário conforme as especificações descritas neste trabalho prático, sem considerar ordenação.
- arquivoSegueOrdenado.bin é o arquivo de dados **segue** no formato binário conforme as especificações descritas neste trabalho prático, considerando a ordenação (ou seja, campo idPessoaQueSegue).

Saída caso o programa seja executado com sucesso:

Listar o arquivo de dados no formato binário usando a função fornecida binarioNaTela.

Mensagem de saída caso algum erro seja encontrado:

Falha no carregamento do arquivo.

Exemplo de execução:

./programaTrab
7 arquivoSegue.bin arquivoSegueOrdenado.bin
usar a função binarioNaTela antes de terminar a execução da funcionalidade, para mostrar a saída do arquivo binário arquivoSegueOrdenado

[8] Estenda a funcionalidade [3] para recuperar os dados armazenados no arquivo de dados **pessoa** associados aos dados no arquivo de dados **segue**, mostrando os dados de forma organizada na saída padrão para permitir a distinção dos campos e registros. O tratamento de 'lixo' deve ser feito de forma a permitir a exibição apropriada dos dados. Registros marcados como logicamente removidos não devem ser exibidos. A funcionalidade [8] equivale a uma operação de junção, desde que ela "junta" dados de registros de dois arquivos usando com base em um campo de igualdade (condição de junção). A junção é uma operação muito importante que vai ser ensinada na disciplina de Bases de Dados.

Existem várias formas de se implementar a junção em bancos de dados. Neste trabalho prático, ela deve ser implementada da seguinte forma. Primeiramente, utilize a funcionalidade [3] para buscar os dados dos registros do arquivo **pessoa** que satisfaçam um critério de busca determinado pelo usuário. Na funcionalidade [8], o usuário somente pode utilizar o campo *idPessoa* como critério de busca. Assim, pode ser recuperado no máximo 1 registro do arquivo de dados **pessoa**. Para esse registro recuperado, utilize o campo *idPessoa* para procurar no arquivo de dados ordenado **segueOrdenado** os registros que satisfaçam à condição de junção de *idPessoa* = *idPessoaQueSegue*. O arquivo **segueOrdenado** encontra-se ordenado pelo campo *idPessoaQueSegue* e, portanto, deve ser utilizada busca binária para recuperar os dados solicitados. As seguintes situações podem ocorrer:

- Não existe igualdade entre *idPessoa* e *idPessoaQueSegue*. Nesse caso, nenhum registro é retornado como resultado da junção.
- Existe igualdade entre *idPessoa* e *idPessoaQueSegue* de forma que existe apenas um registro de **segueOrdenado**. Nesse caso, somente um registro é retornado como resultado da junção.
- Existe igualdade entre *idPessoa* e *idPessoaQueSegue*, de forma que existem vários (ou seja, *n*) registros de **segueOrdenado**. Neste caso, são retornados *n* registros como resultado da junção.

Entrada do programa para a funcionalidade [8]:

`8 arquivoPessoa.bin arquivoIndexaPessoa.bin NomeDoCampo valor arquivoSegueOrdenado.bin`

onde:

- `arquivoPessoa.bin` é o arquivo de dados **pessoa** no formato binário gerado conforme as especificações descritas no primeiro trabalho prático.

- `arquivoIndexaPessoa.bin` é o arquivo de índice **indexaPessoa** gerado conforme as especificações descritas no primeiro trabalho prático.

- `nomeDoCampo` é o nome do campo que está sendo usado na busca. Neste trabalho prático, apenas o campo *idPessoa* pode ser utilizado como critério de busca.

- `valor` é o valor para o campo que está sendo usado na busca.

- `arquivoSegueOrdenado.bin` é o arquivo de dados **segue** no formato binário gerado conforme as especificações descritas neste trabalho prático, o qual encontra-se ordenado de acordo o campo *idPessoaQueSegue*.

Saída caso o programa seja executado com sucesso:

```
A saída deve ser exibida da seguinte forma. Para cada registro do
arquivo arquivoPessoa.bin recuperado na busca feita pela funcionalidade
[3], gere a seguinte saída:
"Dados da pessoa de código " escrever o valor de idPessoa
"Nome: " escrever o valor de nomePessoa
"Idade: " escrever o valor de idadePessoa "anos"
"Twitter: " escrever o valor de twitterPessoa
Deixe uma linha em branco.
Enquanto existirem registros de segueOrdenado tal que idPessoa =
idPessoaQueSegue, gere a seguinte saída:
  "Segue a pessoa de código: " escrever valor de idPessoaQueESeguida
  "Justificativa para seguir: " escrever "segue porque é uma
     celebridade" quando grauAmizade = '0', escrever "segue porque é
     amiga de minha amiga" quando grauAmizade = '1' ou escrever
     "segue porque é minha amiga" quando grauAmizade = '2'
  "Começou a seguir em: " escrever o valor de dataInicioQueSegue
  "Parou de seguir em: " escrever o valor de dataFimQueSegue
  Deixe uma linha em branco
Fim enquanto
Valores nulos devem substituídos pelo caractere "-".
```

Mensagem de saída caso não existam registros:

```
Registro inexistente.
```

Mensagem de saída caso algum erro seja encontrado:

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

Exemplo de execução:

```
./programaTrab
8 arquivoPessoa.bin arquivoIndexaPessoa.bin idPessoa 25
arquivoSegueOrdenado.bin
Dados da pessoa de código 25
Nome: Samantha Pereira Santos
Idade: 13 anos
Twitter: samanthaps
linha em branco
Segue a pessoa de código: 45
Justificativa para seguir: seque porque é uma celebridade
Começou a seguir em: 01/01/2010
Parou de seguir em: 10/05/2015
linha em branco
Segue a pessoa de código: 29
Justificativa para seguir: seque porque é minha amiga
Começou a seguir em: 03/05/2013
Parou de seguir em: 31/12/2020
linha em branco
```

Restrições

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

[1] Os arquivos de dados devem ser gravados 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] É necessário realizar o tratamento de truncamento de dados, conforme as instruções definidas.

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

[6] Os dados devem ser obrigatoriamente escritos campo a campo. Ou seja, não é possível escrever os dados registro a registro. Essa restrição refere-se à entrada/saída, ou seja, à forma como os dados são escritos no arquivo. Para se fazer a busca, é possível caminhar no arquivo registro a registro, já que se sabe o tamanho do 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. Ou seja, 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.
- Um vídeo gravado pelos integrantes do grupo, o qual deve ter por volta de 3 minutos de gravação (um máximo de 5 minutos é permitido, mas o ideal é o vídeo ter por volta de 3 minutos de gravação). O vídeo deve explicar o trabalho desenvolvido. Ou seja, o grupo deve apresentar: cada funcionalidade e uma breve descrição de como a funcionalidade foi implementada. Todos os integrantes do grupo devem participar do vídeo.

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 programaTrab *.c
run:
	./programaTrab
```

Lembrando que *.c já engloba todos os arquivos .c presentes no seu zip. Adicionalmente, no arquivo Makefile é importante se ter um *tab* nos locais colocados acima, senão ele pode não funcionar.

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

- página: https://run.codes/Users/login
- código de matrícula: **V9DB**
- O vídeo gravado deve ser carregado no drive compartilhado SCC0607 Estrutura de Dados III > Trabalhos Práticos > Trabalho Prático T2. O vídeo deve ser nomeado como **Grupo X Video** (onde X representa o número do grupo).

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. A documentação interna terá um peso considerável no trabalho.

- Vídeo. Integrantes que não participarem da apresentação receberão nota 0 no trabalho correspondente.

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 !