Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
Disciplina de Organização de Arquivos (SCC0215)
docente
Profa. Dra. Cristina Dutra de Aguiar Ciferri (cdac@icmc.usp.br)
alunos PAE
Turma A. Raul Donaire (raul.oliveira@usp.br)
Turma B. João Paulo Clarindo (jpcsantos@usp.br)
monitores
Turma A. Vinicius Ricardo Carvalho (vinicius_carvalho@usp.br)
(telegram: @viniciusRC)
Turma B. Mateus Prado Santos (mateus.prado@usp.br)
colaborador
Matheus Carvalho Raimundo (mcarvalhor@usp.br)

Segundo Trabalho Prático

Este trabalho tem como objetivo realizar busca parametrizada, inserção, remoção e atualização de dados armazenados em um arquivo binário.

O trabalho deve ser feito por, no máximo, 2 alunos da mesma turma. Os alunos devem ser o mesmo do Primeiro Tralho Prático. Caso haja mudanças, elas devem ser informadas para a docente, os alunos PAE e 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.

Programa

Descrição Geral. Implemente um programa em C por meio do qual o usuário possa realizar *busca parametrizada*, *inserção*, *remoção* e *atualização* de dados baseado na *abordagem estática* de registros logicamente removidos.

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 Específica. O programa deve oferecer as seguintes funcionalidades:

[3] Permita a recuperação dos dados de todos os registros que satisfaçam um critério de busca determinado pelo usuário, o qual pode ser uma busca combinada. Por exemplo, o usuário pode solicitar a exibição de todos os registros de um determinado *sexoBebe*. Adicionalmente, o usuário pode solicitar uma busca combinada que requeira a exibição de todos os registros de uma determinada *cidadeMae* e um determinado *estadoMae*. Outro exemplo de busca combinada é exibir todos os registros de um determinado *sexoBebe*, de uma determinada *cidadeMae* e de um determinado *estadoMae*. Qualquer combinação de campo pode ser usada como forma de busca. Os valores dos campos do tipo *string* devem ser especificados entre aspas duplas ("). Para a manipulação de *strings* com aspas duplas, pode-se usar a função scan_quote_string disponibilizada na página do projeto da disciplina. Os dados devem ser mostrados no mesmo formato definido para a funcionalidade [2]. Registros marcados como logicamente removidos não devem ser exibidos.

Sintaxe do comando para a funcionalidade [3]:

`3 arquivoGerado.bin m NomeDoCampo`$_1$ `valor`$_1$ `[NomeDoCampo`$_2$ `valor`$_2$ `[...`

onde:

- `arquivoGerado.bin` é o arquivo binário gerado conforme as especificações descritas neste trabalho prático.

- `m` é a quantidade de vezes que `nomeDoCampo` e `valor` podem repetir na sintaxe do comando, devido à busca combinada

- `nomeDoCampo` é o nome do campo que encontra-se definido no arquivo de dados.

- `valor` é o valor utilizado na busca, sendo esse valor diferente de NULO. Os valores dos campos do tipo *string* devem ser especificados entre aspas duplas (").

Saída caso o programa seja executado com sucesso:

```
Podem ser encontrados vários registros que satisfaçam à condição de busca. Apenas alguns dados dos registros de dados devem ser exibidos na saída padrão, a saber:  "Nasceu em " cidadeBebe "/" estadoBebe ", em " dataNascimento ", um bebe de sexo " sexoBebe "." O dado referente ao campo sexoBebe deve ser escrito da seguinte forma: "IGNORADO" caso sexoBebe = ‘0’, "MASCULINO" caso sexoBebe = ‘1’ e "FEMININO" caso sexoBebe = ‘2’. Campos que tiverem o valor nulo não devem ser mostrados, sendo substituídos pelo caractere "-". Para a manipulação de strings com aspas duplas, pode-se usar a função scan_quote_string disponibilizada na página da disciplina.
```

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 (é mostrado apenas um registro, embora a funcionalidade provida pelo programa deva exibir mais do que um registro quando for o caso; o registro é apenas ilustrativo)**:**

```
./programaTrab
3 arquivoGerado.bin 2 estadoBebe “SP” sexo “2”
Nasceu em SAO CARLOS/SP, em 2020-04-18, um bebe de sexo FEMININO.
```

[4] Permita a recuperação dos dados de um registro, a partir da identificação do RRN (número relativo do registro) do registro desejado pelo usuário. Por exemplo, o usuário pode solicitar a recuperação dos dados do registro de RRN = 2 ou do registro de RRN = 4. Os dados solicitados devem ser mostrados no mesmo formato definido para a funcionalidade [2]. Registros marcados como logicamente removidos não devem ser exibidos.

Sintaxe do comando para a funcionalidade [4]:

```
4 arquivoGerado.bin  RRN
```

Saída caso o programa seja executado com sucesso:

Será recuperado, no máximo, 1 registro. Apenas alguns dados dos registros de dados devem ser exibidos na saída padrão, a saber: "Nasceu em " cidadeBebe "/" estadoBebe ", em " dataNascimento ", um bebe de sexo " sexoBebe "." O dado referente ao campo sexoBebe deve ser escrito da seguinte forma: "IGNORADO" caso sexoBebe = '0', "MASCULINO" caso sexoBebe = '1' e "FEMININO" caso sexoBebe = '2'. Campos que tiverem o valor nulo não devem ser mostrados, sendo substituídos pelo caractere "-".

Mensagem de saída caso não seja encontrado o registro ou o registro esteja removido:

```
Registro inexistente.
```

Mensagem de saída caso algum erro seja encontrado:

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

Exemplo de execução:

```
./programaTrab
4 arquivoGerado.bin 1
Nasceu em ARARAQUARA/SP, em -, um bebe do sexo IGNORADO.
```

[5] Permita a remoção lógica de registros, baseado na *abordagem estática* de registros logicamente removidos. A implementação dessa funcionalidade deve seguir estritamente a matéria apresentada em sala de aula. A marcação dos registros logicamente removidos deve ser feita armazenando-se -1 no primeiro campo do registro, que é um campo do tipo inteiro. Os demais caracteres devem permanecer como estavam anteriormente. Os registros a serem removidos devem ser aqueles que satisfaçam um critério de busca (ou busca combinada) determinado pelo usuário, conforme detalhado na funcionalidade [3]. Não deve ser realizado o tratamento da fragmentação interna. Também não deve ser realizado o tratamento de fragmentação externa usando a técnica de coalescimento. A funcionalidade [5] deve ser executada *n* vezes seguidas. Em situações nas quais um determinado critério de busca não seja satisfeito, ou seja, caso a solicitação do usuário não retorne nenhum registro a ser removido, o programa deve continuar a executar as remoções até completar as *n* vezes seguidas. 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. Registros que já foram marcados anteriormente como logicamente removidos não podem ser marcados como logicamente removidos novamente. Lembre-se de manipular o campo “status” do registro de cabeçalho adequadamente. Quando o arquivo começar a ser modificado, deve ser marcado como inconsistente. Ao final de todas as modificações, deve ser marcado como consistente.

Entrada do programa para a funcionalidade [5]:

`5 arquivoGerado.bin n`

m_1 `nomeCampo`$_{11}$ `valorCampo`$_{11}$ [`nomeCampo`$_{12}$ `valorCampo`$_{12}$ [...

m_2 `nomeCampo`$_{21}$ `valorCampo`$_{21}$ [`nomeCampo`$_{22}$ `valorCampo`$_{22}$ [...

...

m_n `nomeCampo`$_{n1}$ `valorCampo`$_{n1}$ [`nomeCampo`$_{n2}$ `valorCampo`$_{n2}$ [...

onde:

- `arquivoGerado.bin` é um arquivo binário de entrada que segue as especificações definidas neste trabalho prático. As remoções a serem realizadas nessa funcionalidade devem ser feitas nesse arquivo.
- `n` é o número de remoções a serem realizadas. Para cada remoção, deve ser informado o `nome do campo` a ser considerado e seu critério de busca representado pelo `valor do campo`, de acordo com as especificações feitas na funcionalidade [3].
- `m` é o número de vezes que `nomeCampo` e `valorCampo` podem repetir na sintaxe do comando, devido à busca combinada.

Cada uma das `n` remoções deve ser especificada em uma linha diferente. Deve ser deixado um espaço em branco entre o `nome do campo` e o `valor do campo`. Os valores dos campos do tipo *string* devem ser especificados entre aspas duplas ("). Para a manipulação de strings com aspas duplas, pode-se usar a função scan_quote_string disponibilizada na página do projeto da disciplina.

Saída caso o programa seja executado com sucesso:

`Listar o arquivo binário arquivoGerado.bin usando a função binarioNaTela.`

Mensagem de saída caso algum erro seja encontrado:

`Falha no processamento do arquivo.`

Exemplo de execução:

```
./programaTrab
5 arquivoGerado.bin 2
2 estadoBebe “SP” sexo “2”
1 cidadeBebe “SAO PAULO”
```

`usar a função binarioNaTela antes de terminar a execução da funcionalidade, para mostrar a saída do arquivo arquivoGerado.bin, o qual foi atualizado com as remoções.`

[6] Permita a inserção de registros adicionais, baseado na *abordagem estática* de registros logicamente removidos. A implementação dessa funcionalidade deve seguir estritamente a matéria apresentada em sala de aula. Não é necessário realizar o tratamento de truncamento de dados. Portanto, a soma dos tamanhos para os campos de tamanho variável nunca deve ultrapassar o tamanho do registro de tamanho fixo. Campos com valores nulos, na entrada da funcionalidade, devem ser identificados com NULO. A funcionalidade [6] deve ser executada *n* vezes seguidas. 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. Quando o arquivo começar a ser modificado, deve ser marcado como inconsistente. Ao final de todas as modificações, deve ser marcado como consistente.

Entrada do programa para a funcionalidade [6]:

`6 arquivoGerado.bin n`

`cidadeMae`$_1$ `cidadeBebe`$_1$ `idNascimento`$_1$ `idadeMae`$_1$ `dataNascimento`$_1$ `sexoBebe`$_1$ `estadoMae`$_1$ `estadoBebe`$_1$

`cidadeMae`$_2$ `cidadeBebe`$_2$ `idNascimento`$_2$ `idadeMae`$_2$ `dataNascimento`$_2$ `sexoBebe`$_2$ `estadoMae`$_2$ `estadoBebe`$_2$

`...`

`cidadeMae`$_n$ `cidadeBebe`$_n$ `idNascimento`$_n$ `idadeMae`$_n$ `dataNascimento`$_n$ `sexoBebe`$_n$ `estadoMae`$_n$ `estadoBebe`$_n$

onde:

- `arquivoGerado.bin` é um arquivo binário de entrada que segue as mesmas especificações definidas nesse trabalho prático. As inserções a serem realizadas nessa funcionalidade devem ser feitas nesse arquivo.
- `n` é o número de inserções a serem realizadas. Para cada inserção, deve ser informado os valores a serem inseridos no arquivo, para os campos especificados na mesma ordem que a definida nesse trabalho prático, a saber: `cidadeMae`, `cidadeBebe`, `idNascimento`, `idadeMae`, `dataNascimento`, `sexoBebe`, `estadoMae`, `estadoBebe`. Não existe truncamento de dados. Valores nulos devem ser identificados, na entrada da funcionalidade, por NULO. Cada uma das `n` inserções deve ser especificada em uma linha diferente. Deve ser deixado um espaço em branco entre os valores dos campos. Os valores dos campos do tipo *string* devem ser especificados entre aspas duplas (").

Saída caso o programa seja executado com sucesso:

`Listar o arquivo binário arquivoGerado.bin usando a função binarioNaTela.`

Mensagem de saída caso algum erro seja encontrado:

`Falha no processamento do arquivo.`

Exemplo de execução:

```
./programaTrab
6 arquivoGerado.bin 2
“MATAO“ “RIBEIRAO PRETO“ 3 28 “2019-05-20“ “2“ “SP” “SP”
“ARARAQUARA“ “ARARAQUARA“ 5 NULO NULO “1“ “SP” “SP”
```

`usar a função binarioNaTela antes de terminar a execução da funcionalidade, para mostrar a saída do arquivo arquivoGerado.bin, o qual foi atualizado com as inserções.`

[7] Permita a atualização de um ou mais campos de um registro identificado por seu RRN. Não é necessário realizar o tratamento de truncamento de dados. Portanto, a soma dos tamanhos para os campos de tamanho variável nunca deve ultrapassar o tamanho do registro de tamanho fixo. O lixo no registro atualizado, quando ele tiver tamanho menor do que o registro antes de ser atualizado, deve permanecer com os caracteres que já estavam anteriormente. Campos a serem atualizados com valores nulos devem ser identificados, na entrada da funcionalidade, com NULO. A funcionalidade [7] deve ser executada *n* vezes seguidas. Em situações nas quais um determinado RRN não seja encontrado, ou seja, caso a solicitação do usuário não retorne nenhum registro a ser atualizado, o programa deve continuar a executar as atualizações até completar as *n* vezes seguidas. 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. Quando o arquivo começar a ser modificado, deve ser marcado como inconsistente. Ao final de todas as modificações, deve ser marcado como consistente.

Entrada do programa para a funcionalidade [7]:

`7 arquivoGerado.bin n`

`RRN` m_1 `nomeCampo`$_{11}$ `valorCampo`$_{11}$ `[nomeCampo`$_{12}$ `valorCampo`$_{12}$ `[...`
`RRN` m_2 `nomeCampo`$_{21}$ `valorCampo`$_{21}$ `[nomeCampo`$_{22}$ `valorCampo`$_{22}$ `[...`
`...`
`RRN` m_n `nomeCampo`$_{n1}$ `valorCampo`$_{n1}$ `[nomeCampo`$_{n2}$ `valorCampo`$_{n2}$ `[...`

onde:

- `arquivoGerado.bin` é um arquivo binário de entrada que segue as mesmas especificações definidas no primeiro trabalho prático. As atualizações a serem realizadas nessa funcionalidade devem ser feitas nesse arquivo.
- `n` é o número de atualizações a serem realizadas. Para cada atualização, deve ser informado o valor do RRN do registro, os campos do registro a serem alterados e os novos valores desses campos. Não existe truncamento de dados. Valores nulos devem ser identificados, na entrada da funcionalidade, por NULO.
- `m` é o número de vezes que `nomeCampo` e `valorCampo` podem repetir na sintaxe do comando, desde que vários campos de um mesmo registro podem ser atualizados de uma única vez.

Cada uma das `n` atualizações deve ser especificada em uma linha diferente. Deve ser deixado um espaço em branco entre o `RRN` e o `nome do campo`, e entre o `nome do campo` e o `valor do campo`. Os valores dos campos do tipo *string* devem ser especificados entre aspas duplas ("). Para a manipulação de strings com aspas duplas, pode-se usar a função scan_quote_string disponibilizada na página do projeto da disciplina.

Saída caso o programa seja executado com sucesso:

`Listar o arquivo binário arquivoGerado.bin usando a função binarioNaTela.`

Mensagem de saída caso algum erro seja encontrado:

`Falha no processamento do arquivo.`

Exemplo de execução:

```
./programaTrab
7 arquivoGerado.bin 2
3 1 cidadeMae "GUARULHOS"
5 2 cidadeMae "SALVADOR" dataNascimento NULO
usar a função binarioNaTela antes de terminar a execução da funcionalidade, para mostrar a saída do arquivo arquivoGerado.bin, o qual foi atualizado com as atualizações
```

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 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.

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
- **Turma A** (segunda-feira): código de matrícula: **SBRZ**
- **Turma B** (terça-feira): código de matrícula: **24AK**

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.

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 !