**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)

# Primeiro Trabalho Prático

**Este trabalho tem como objetivo armazenar dados em um arquivo binário de acordo com uma organização de campos e registros, bem como recuperar todos os dados armazenados.**

*O trabalho deve ser feito por, no máximo, 2 alunos da mesma turma. 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 do arquivo de dados

---

**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.
- *RRNproxRegistro*: indica o RRN do próximo registro a ser inserido. Quando o arquivo é criado, *RRNproxRegistro* = 0. Depois disso, *RRNproxRegistro* é incrementado cada vez que um registro é inserido. – tamanho: inteiro de 4 bytes. Note que o campo *RRNproxRegistro* deve ser utilizado sempre que um novo

registro for inserido para calcular o *byte offset* no qual o registro deve ser armazenado.

- *numeroRegistrosInseridos*: indica o número de registros que foram inseridos no arquivo. Quando o arquivo é criado, *numeroRegistrosInseridos* = 0. Depois que o arquivo é carregado com os dados do arquivo .csv, *numeroRegistrosInseridos* deve ser atualizado para armazenar a quantidade de registros que foram inseridos. Depois, *numeroRegistrosInseridos* é incrementado a cada vez que um registro é inserido no arquivo de dados – tamanho: inteiro de 4 bytes. Note que o valor do campo *numeroRegistrosInseridos* incrementa quando novos registros estão sendo inseridos, e decrementa quando registros já existentes são marcados como logicamente removidos.
- *numeroRegistrosRemovidos*: indica o número de registros que estão marcados como removidos. Quando o arquivo é criado, *numeroRegistrosRemovidos* = 0. Depois disso, *numeroRegistrosRemovidos* é incrementado a cada vez que um registro é marcado como logicamente removido – tamanho: inteiro de 4 bytes. Note que o valor do campo *numeroRegistrosRemovidos* incrementa quando um registro já existente é marcado como removido.
- *numeroRegistrosAtualizados*: indica o número de registros do arquivo de dados que foram atualizados. Quando o arquivo é criado, *numeroRegistrosAtualizados* = 0. Depois disso, *numeroRegistrosAtualizados* é incrementado a cada vez que um registro é atualizado – tamanho: inteiro de 4 bytes. Note que o valor do campo *numeroRegistrosAtualizados* incrementa quando um registro já existente é atualizado.

**Representação Gráfica do Registro de Cabeçalho.** O tamanho de cada registro de dados deve ser de 128 bytes, representado da seguinte forma:

| 1 byte | 4 bytes | | | | 4 bytes | | | | 4 bytes | | | | 4 bytes | | | | 111 bytes | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *status* | *RRNprox Registro* | | | | *numeroRegistros Inseridos* | | | | *numeroRegistros Removidos* | | | | *numeroRegistros Atualizados* | | | | *lixo (caractere '$')* | |
| *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | *13* | *14* | *15* | *16* | *...* | *127* |

**Observações Importantes.**

- O registro de cabeçalho 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.
- Para caber em páginas de disco (que são definidas em potência de 2), o registro de cabeçalho também foi definido como uma potência de 2, no mesmo tamanho dos registros de dados). Portanto, o registro de cabeçalho tem o tamanho de 128 bytes, sendo 17 bytes preenchidos com dados necessários para o desenvolvimento do trabalho, e os 111 bytes restantes preenchidos com lixo. O lixo é representado pelo caractere '$'.

---

**Registros de Dados.** Deve ser considerada a *organização híbrida de campos e registros*, da seguinte forma:

- Campos de tamanho fixo e campos de tamanho variável. Para os campos de tamanho variável, deve-se usar o método *indicador de tamanho* (número inteiro de 4 bytes).
- Registros de tamanho fixo.

**Observação.** Cuidado ao definir a organização do arquivo de dados. Analise os slides com título "Organização híbrida de campos e registros" disponíveis no arquivo http://wiki.icmc.usp.br/images/3/33/SCC0215012018camposRegistros.pdf.

---

**Descrição dos Registros de Dados.**

Cada registro do arquivo de dados deve conter dados relacionados aos nascidos vivos, os quais são disponibilizados pelo sistema SINASC (Sistema de Informações sobre Nascidos Vivos) do DATASUS. Os dados utilizados neste trabalho prático foram obtidos a partir dos resultados divulgados no artigo científico: CLARINDO, J.P.; FONTES, W.S.; COUTINHO, F. QualiSUS: um *dataset* sobre dados da Saúde Pública

no Brasil. In Proceedings Companion of the 34rd Brazilian Symposium on Databases, p. 418-428, 2019.

- Campos de tamanho fixo: tamanho de 23 bytes
  - *idNascimento* – inteiro – tamanho: 4 bytes (código sequencial que identifica univocamente cada registro do arquivo de dados).
  - *idadeMae* – inteiro – tamanho: 4 bytes (armazena a idade da mãe do bebê).
  - *dataNascimento* – *string* – tamanho: 10 bytes, no formato AAAA-MM-DD (armazena a data de nascimento do bebê).
  - *sexoBebe* – string – tamanho: 1 byte (armazena o sexo do bebê) – pode assumir os valores '0' (ignorado), '1' (masculino) e '2' (feminino).
  - *estadoMae* – *string* – tamanho: 2 bytes (armazena a sigla do estado da cidade de residência da mãe).
  - *estadoBebe* – *string* – tamanho: 2 bytes (armazena a sigla do estado da cidade na qual o bebê nasceu).
- Campos de tamanho variável: tamanho máximo de 104 bytes, incluindo os espaços reservados para os indicadores de tamanho
  - *cidadeMae* – *string* de tamanho variável (armazena o nome da cidade de residência da mãe).
  - *cidadeBebe* – *string* de tamanho variável (armazena o nome da cidade na qual o bebê nasceu).

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

**Representação Gráfica do Registro de Dados.** O tamanho de cada registro de dados deve ser de 128 bytes, representado da seguinte forma:

| 4 bytes | 4 bytes | variável | variável | 4 bytes | 4 bytes | 10 bytes | 1 byte | 2 bytes | 2 bytes |
|---|---|---|---|---|---|---|---|---|---|
| *campo 1* | *campo 2* | *campo 3* | *campo 4* | *campo 5* | *campo 6* | *campo 7* | *campo 8* | *campo 9* | *campo 10* |
| 0 1 2 3 | 4 5 6 7 | ... ... | | 105...108 | 109...112 | 113...122 | 123 | 124 125 | 126 127 |

*campo1. tamanho do campo cidadeMae (tamanho fixo de 4 bytes)*
*campo2. tamanho do campo cidadeBebe (tamanho fixo de 4 bytes)*
*campo3. cidadeMae (tamanho variável)*
*campo4. cidadeBebe (tamanho variável)*
*campo5. idNascimento (tamanho fixo de 4 bytes)*
*campo6. idadeMae (tamanho fixo de 4 bytes)*
*campo7. dataNascimento (tamanho fixo de 10 bytes)*
*campo8. sexoBebe (tamanho fixo de 1 byte)*
*campo9. estadoMae (tamanho fixo de 2 bytes)*
*campo10. estadoBebe (tamanho fixo de 2 bytes)*
*em azul. representação de byte offset, indicando que os campos de tamanho fixo começam no byte offset 105 do registro*

**Observações sobre a organização dos registros de dados.** Neste projeto, optou-se por organizar os registros de dados da seguinte forma. Primeiro, tem-se os campos de tamanho variável. Esses campos são representados usando o método indicador de tamanho. Optou-se por armazenar, em *campo1* o tamanho da cidade da mãe do bebê e em *campo2* o tamanho da cidade do bebê. Na sequencia, aparecem *campo3* e *campo4*, que contêm a cidade da mãe do bebê e a cidade na qual o bebê nasceu. No final do registro, aparecem os campos de tamanho fixo. São representados o código sequencial que identifica univocamente cada registro do arquivo de dados (*campo5*), a idade da mãe (*campo6*), a data de nascimento do bebê (*campo7*), o sexo do bebê (*campo8*), a sigla do estado da cidade de residência da mãe (*campo9*) e a sigla do estado da cidade na qual o bebê nasceu (*campo10*).

**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.
- Deve ser feita a diferenciação entre o espaço utilizado e o lixo. Para tanto, todas as *strings* de tamanho variável devem ser finalizadas com '\0' e o lixo deve ser identificado pelo caractere '$'. Nenhum *byte* do registro deve permanecer vazio, ou seja, cada *byte* deve armazenar um valor válido ou '\0' ou '$'.
- *Strings* de tamanho fixo não devem ser finalizadas com '\0'.
- Apenas o campo *idNascimento* não aceita valores nulos. O arquivo .csv com os dados de entrada já garante essa característica.
- Para os campos de tamanho fixo, os valores nulos devem ser representados da seguinte forma:
    - se o campo é inteiro ou de dupla precisão, então armazena-se o valor -1
    - se o campo é do tipo *string*, então armazena-se '\0$$$$$$$$$$'
- Para os campos de tamanho variável, os valores nulos devem ser representados da seguinte forma:
    - deve ser armazenado apenas o indicador de tamanho do campo, o qual deve possuir o valor 0.
- Não é necessário realizar o tratamento de truncamento de dados. O arquivo .csv com os dados de entrada já garante essa característica.
- **Campo1** é um campo do tipo inteiro que é utilizado para indicar o tamanho do campo cidadeMae. Ele também deve ser usado para representar que um registro encontra-se logicamente removido. A remoção lógica de registros será baseada na abordagem estática, e será definida em detalhes no **Segundo Trabalho Prático**. Note, entretanto, que neste primeiro trabalho prático já se deve verificar se um registro encontra-se logicamente removido ou não. Registros logicamente removidos são identificados pelo inteiro -1 armazenado em **Campo1**.

## Programa

**Descrição Geral**. Implemente um programa em C por meio do qual o usuário possa obter dados de um arquivo de entrada e gerar um arquivo binário com esses dados, bem como listar todos os dados contidos nesse arquivo binário.

**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:

[1] 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 que o arquivo de dados de saída deve ser gerado como parte deste trabalho prático. 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 [1]:**

`1 arquivoEntrada.csv  arquivoGerado.bin`

**onde:**

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

**Saída caso o programa seja executado com sucesso:**

`Listar o arquivo de saída 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
1 arquivoEntrada.csv arquivoGerado.bin
usar a função binarioNaTela antes de terminar a execução da funcionalidade, para mostrar a saída do arquivo arquivoGerado.bin
```

[2] Permita a recuperação dos dados, de todos os registros, armazenados no arquivo de dados, 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.

**Entrada do programa para a funcionalidade [2]:**

```
2 arquivoGerado.bin
```

**onde:**

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

**Saída caso o programa seja executado com sucesso:**

```
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 existam registros:**

```
Registro inexistente.
```

**Mensagem de saída caso algum erro seja encontrado:**

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

**Exemplo de execução** (são mostrados alguns registros ilustrativos)**:**

```
./programaTrab
2 arquivoGerado.bin
Nasceu em SAO CARLOS/SP, em 2020-04-18, um bebe de sexo FEMININO.
Nasceu em ARARAQUARA/SP, em -, um bebe do sexo IGNORADO.
Nasceu em SAO PAULO/-, em 2020-03-13, um bebe do sexo MASCULINO.
...
```

## 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 !**