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]

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 os dados armazenados.

*O trabalho deve ser feito **individualmente**. A solução deve ser proposta exclusivamente pelo aluno com base nos conhecimentos adquiridos nas aulas. Consulte as notas de aula e o livro texto quando necessário.*

Descrição de páginas de disco

No trabalho será usado o conceito de páginas de disco. Cada página de disco tem o tamanho fixo de 16.000 bytes.

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.
- *topoPilha*: armazena o RRN de um registro logicamente removido, ou -1 caso não haja registros logicamente removidos – tamanho: inteiro de 4 bytes.

- *tagCampo1*: valor resumido da *tag* para o campo *nroInscricao*. Deve assumir o valor 1 – tamanho: *string* de 1 byte.
- *desCampo1*: descrição completa do campo *nroInscricao*. Deve assumir o valor 'numero de inscricao do participante do ENEM' – tamanho: *string* de 55 bytes.
- *tagCampo2*: valor resumido da *tag* para o campo *nota*. Deve assumir o valor 2 – tamanho: *string* de 1 byte.
- *desCampo2*: descrição completa do campo *nota*. Deve assumir o valor 'nota do participante do ENEM na prova de matematica' – tamanho: *string* de 55 bytes.
- *tagCampo3*: valor resumido da *tag* para o campo *data*. Deve assumir o valor 3 – tamanho: *string* de 1 byte.
- *desCampo3*: descrição completa do campo data. Deve assumir o valor 'data' – tamanho: *string* de 55 bytes.
- *tagCampo4*: valor resumido da *tag* para o campo *cidade*. Deve assumir o valor 4 – tamanho: *string* de 55 bytes.
- *desCampo4*: descrição completa do campo *cidade*. Deve assumir o valor 'cidade na qual o participante do ENEM mora' – tamanho: *string* de 55 bytes.
- *tagCampo5*: valor resumido da *tag* para o campo *nomeEscola*. Deve assumir o valor 5 – tamanho: *string* de 1 byte.
- *desCampo5*: descrição completa do campo *nomeEscola*. Deve assumir o valor 'nome da escola de ensino medio' – tamanho: *string* de 55 bytes.

Representação Gráfica do Registro de Cabeçalho. O tamanho do registro de cabeçalho deve ser de 284 bytes, representado da seguinte forma:

1 byte	4 bytes				1 byte	55 bytes	1 byte	55 bytes	1 byte	55 bytes	1 byte	55 bytes	1 byte	55 bytes	1 byte	55 bytes
status	*topoPilha*				*tag Campo1*	*des Campo1*	*tag Campo2*	*des Campo2*	*tag Campo3*	*des Campo3*	*tag Campo4*	*des Campo4*	*tag Campo5*	*des Campo5*		
0	1	2	3	4	.....											284

Página de disco. O registro de cabeçalho deve ocupar uma página de disco.

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.
- Deve ser feita a diferenciação entre o espaço utilizado e o lixo. Para tanto, todas as *strings* devem ser finalizadas com '\0' e o lixo deve ser identificado 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*.
- 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 participantes do ENEM. Esses dados foram gerados usando dados reais obtidos do ENEM e também dados sintéticos. Cada registro representa um participante do ENEM e contém os seguintes campos:

- Campos de tamanho fixo: 22 bytes
  - *nroInscricao* – inteiro – tamanho: 4 bytes
  - *nota* – número de dupla precisão – tamanho: 8 bytes
  - *data* – tamanho: 10 bytes, no formato DD/MM/AAAA
- Campos de tamanho variável:
  - *cidade* – *string* de tamanho variável
  - *nomeEscola* – *string* de tamanho variável

Os dados dos participantes do ENEM são fornecidos juntamente com a especificação deste trabalho prático por meio de um arquivo .csv, sendo que sendo que

sua especificação encontra-se disponível na página da disciplina. No arquivo .csv, o separador de campos é vírgula (,).

Além de armazenar os dados relacionados aos participantes do ENEM, cada registro de dados conterá campos adicionais para a implementações futuras relacionadas à remoção de registros logicamente removidos. Esses campos são:

- *removido*: indica se o registro se encontra removido ou não. Pode assumir os valores '*', para indicar que o registro é um registro removido, ou '-', para indicar que o registro não é um registro removido – tamanho: *string* de 1 byte
- *encadeamento*: armazena os RRNs dos registros logicamente removidos – tamanho: inteiro de 4 bytes.

Representação Gráfica do Registro de Dados. O tamanho de cada registro de dados deve ser de 80 bytes, representado da seguinte forma:

1 byte	4 bytes				4 bytes	8 bytes	10 bytes	4 bytes	1 byte	tamanho variável	4 bytes	1 byte	tamanho variável
removido	*encadeamento*				*nro Inscricao*	*nota*	*data*	indicador de tamanho	*tagCampo4 (deve ter o valor 4)*	*cidade*	indicador de tamanho	*tagCampo5 (deve ter o valor 5)*	*nomeEscola*
0	1	2	3	4	.....								79

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* devem ser finalizadas com '\0' e o lixo deve ser identificado pelo caractere '@'. Ou seja, quando sobra-se espaço no final do registro, o registro deve ser completado com lixo até o seu final.
- O campo *nroInscricao* não aceita valores repetidos e nem valores nulos.
- Os campos *nota*, *data*, *cidade* e *nomeEscola* aceitam valores repetidos e valores nulos.

- 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:
    - não devem ser armazenados os campos referentes: (i) ao indicador de tamanho; (ii) à *tag* que representa o dado; e (iii) ao valor do dado.
- Não é necessário realizar o tratamento de truncamento de dados.
- Neste primeiro trabalho prático, os campos *removido* e *encadeamento* não serão utilizados. Para cada registro, esses campos devem ser inicializados da seguinte forma: (i) *removido* deve ser inicializado com o valor '-'; e (ii) *encadeamento* deve ser inicializado com o valor -1.

Página de disco. Os registros de dados não devem ser armazenados na mesma página de disco que o registro de cabeçalho. Adicionalmente, os registros de dados devem ser armazenados em várias páginas de disco, de acordo com a quantidade de registros gerados.

Programa

Descrição Geral. Implemente um programa em C que ofereça uma interface, via linha de comando, por meio da qual o usuário possa obter dados de um arquivo de entrada e gerar um arquivo binário com esses dados, bem como exibir os dados armazenados no arquivo binário. Deve-se levar em consideração a descrição e a organização do arquivo de dados especificados anteriormente. A definição da sintaxe de cada comando deve seguir estritamente as especificações definidas em cada funcionalidade.

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 "programaTrab1". Essas orientações devem ser seguidas uma vez que a correção do

funcionamento do programa se dará de forma automática. De forma geral, o primeiro parâmetro a ser passado para o programa por meio da linha de comando 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 será fornecido juntamente com a especificação do projeto, enquanto que o arquivo de dados de saída deverá ser gerado como parte deste trabalho prático.

Entrada do programa para a funcionalidade [1]:

```
1 arquivo.csv  // arquivo de entrada no formato .csv
```

Saída caso o programa seja executado com sucesso:

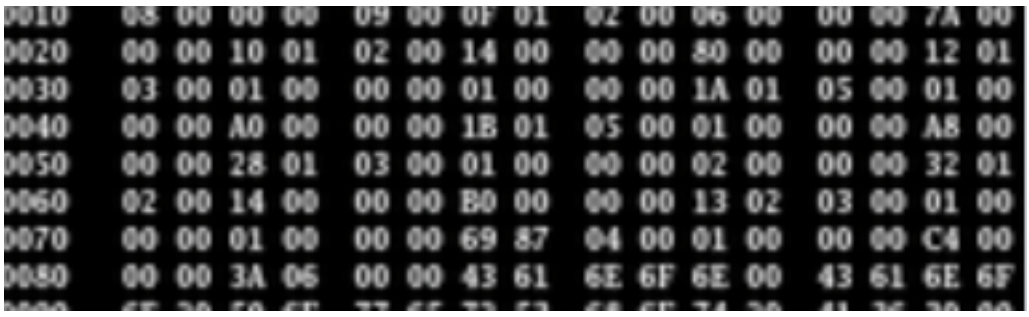
```
Listar na saída padrão o arquivo binário gerado.
```

Mensagem de saída caso algum erro seja encontrado:

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

Exemplo de execução:

```
./programaTrab1
1 arquivo.csv
```



[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. Depois de mostrar todos os registros, deve ser mostrado na saída padrão o número de páginas de disco acessadas.

Entrada do programa para a funcionalidade [2]:

```
2 arquivo.bin   //arquivo binário gerado na funcionalidade [1]
```

Saída caso o programa seja executado com sucesso:

Cada 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 seu tamanho em bytes. Para os campos de tamanho variável com valores nulos, não deve ser exibido nada. Ao final, deve ser exibido o número de páginas de disco acessadas.

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 apenas 2 registros)**:**

```
./programaTrab1
2 arquivo.bin
439 607.5 01/01/2004 6 Maceio 8 PEDRO II
387 9 Sao Paulo 10 JOAO KOPKE
Número de páginas de disco acessadas: 25
```

[3] Permita a recuperação dos dados de todos os registros que satisfaçam um critério de busca determinado pelo usuário. Por exemplo, o usuário pode solicitar a exibição de todos os registros de um determinado *número de inscrição*. Note que qualquer campo pode ser utilizado como forma de busca. Os dados solicitados devem ser mostrados no mesmo formato definido para a funcionalidade [2]. Depois de mostrar todos os registros, deve ser mostrado na saída padrão o número de páginas de disco acessadas.

Sintaxe do comando para a funcionalidade [3]:

```
3 arquivo.bin NomeDoCampo valor
```

Saída caso o programa seja executado com sucesso:

```
Podem ser encontrados vários registros que satisfaçam à condição de busca. Cada 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 seu tamanho em bytes. Para os campos de tamanho variável com valores nulos, não deve ser exibido nada. Ao final, deve ser exibido o número de páginas de disco acessadas.
```

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 o primeiro registro que satisfez à busca, embora a funcionalidade provida pelo programa deva exibir mais do que um registro quando for o caso)**:**

```
./programaTrab1
3 nroInscricao 332
332 400.8 03/01/2004 8 Brasilia 29 REINALDO RIBEIRO DA SILVA DOU
Número de páginas de disco acessadas: 1
```

[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]. Depois de mostrar todos os registros, deve ser mostrado na saída padrão o número de páginas de disco acessadas.

Sintaxe do comando para a funcionalidade [3]:

```
3 arquivo.bin  RRN
```

Saída caso o programa seja executado com sucesso:

Será recuperado, no máximo, 1 registro. 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 seu tamanho em bytes. Para os campos de tamanho variável com valores nulos, não deve ser exibido nada. Ao final, deve ser exibido o número de páginas de disco acessadas.

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:

```
./programaTrab1
4 arquivo.bin 1
387 9 Sao Paulo 10 JOAO KOPKE
Número de páginas de disco acessadas: 1
```

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] Os integrantes do grupo devem constar como comentário no início do código (i.e. NUSP e nome de cada integrante do grupo). 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 interface deve obrigatoriamente ser via linha de comando, sendo que a sintaxe de cada comando deve seguir estritamente as especificações definidas em cada funcionalidade.

[10] 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 nas transparências 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 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 !