Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
Disciplina de Organização de Arquivos
docente
Profa. Dra. Cristina Dutra de Aguiar Ciferri (cdac@icmc.usp.br)
alunos PAE
Turma A. Viviana Elizabeth Romero Noguera (viviana.noguera@usp.br)
Turma B. Guilherme Muzzi da Rocha (guilherme.muzzi.rocha@usp.br)
monitores
Turmas A e B. Matheus Carvalho Raimundo (mcarvalhor@usp.br)
Turma B. Gabriel Alfonso Nascimento Salgueiro (gabrielsalgueiro@usp.br)

Segundo Trabalho Prático

Este trabalho tem como objetivo realizar operações de inserção, remoção e atualização de dados baseadas na abordagem dinâmica de reaproveitamento de espaços de registros logicamente removidos.

*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.*

Programa

Descrição Geral. Implemente um programa em C que ofereça uma interface por meio da qual o usuário possa realizar *inserção*, *remoção* e *atualização* de dados baseado na *abordagem dinâmica* de reaproveitamento de espaços de registros logicamente removidos. Deve-se levar em consideração a descrição e a organização do arquivo de dados especificados no primeiro trabalho prático.

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 "programaTrab2". 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:

[4] Permita a remoção lógica de registros, baseado na *abordagem dinâmica* de reaproveitamento de espaços de registros logicamente removidos. A implementação dessa funcionalidade deve ser realizada usando o conceito de *lista ordenada de forma crescente em termos do tamanho do registro*, e deve seguir estritamente a matéria apresentada em sala de aula. Os registros a serem removidos devem ser aqueles que satisfaçam um critério de busca determinado pelo usuário. Por exemplo, o usuário pode solicitar a remoção de um registro que possui um determinado *identificador do servidor* ou o usuário pode solicitar a remoção de todos os registros de um determinado *cargoServidor*. Note que qualquer campo pode ser utilizado como forma de remoção. Não deve ser realizado o tratamento da fragmentação interna, sendo que o lixo que permanece no registro logicamente removido deve ser identificado pelo caractere @. Também não deve ser realizado o tratamento de fragmentação externa usando a técnica de coalescimento. A funcionalidade [4] 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 escreverNaTela1 ou escreverNaTela2, ambas disponibilizadas na página do projeto da disciplina, para mostrar a saída do arquivo binário.

Entrada do programa para a funcionalidade [4]:

```
4 arquivo.bin n
nomeCampo1 valorCampo1
nomeCampo2 valorCampo2
...
nomeCampon valorCampon
```

onde:

- `arquivo.bin` é um arquivo binário de entrada que segue as mesmas especificações definidas no primeiro 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`. 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 (").

Saída caso o programa seja executado com sucesso:

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

Mensagem de saída caso algum erro seja encontrado:

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

Exemplo de execução:

```
./programaTrab2
4 arquivo.bin 2
idServidor 1914208
cargoServidor "TECNICO EM SAUDE PUBLICA"
usar a função escreverNaTela1 ou escreverNaTela2 antes de terminar a execução da funcionalidade, para mostrar a saída do arquivo arquivo.bin, o qual foi atualizado com as remoções.
```

[5] Permita a inserção de registros adicionais, baseado na *abordagem dinâmica* de reaproveitamento de espaços de registros logicamente removidos. A implementação dessa funcionalidade deve ser realizada usando o conceito de *lista ordenada de forma crescente em termos do tamanho do registro*, e deve seguir estritamente a matéria apresentada em sala de aula. O lixo que permanece no registro logicamente removido deve ser identificado pelo caractere @. Campos com valores nulos devem ser identificados, na entrada da funcionalidade, com NULO. A funcionalidade [5] deve ser executada *n* vezes seguidas. Antes de terminar a execução da funcionalidade, deve ser utilizada a função escreverNaTela1 ou escreverNaTela2, ambas disponibilizadas na página do projeto da disciplina, para mostrar a saída do arquivo binário.

Entrada do programa para a funcionalidade [5]:

`5 arquivo.bin n`
`valorIdServidor`$_1$ `valorSalarioServidor`$_1$ `valorTelefoneServidor`$_1$
`valorNomeServidor`$_1$ `valorCargoServidor`$_1$
`valorIdServidor`$_2$ `valorSalarioServidor`$_2$ `valorTelefoneServidor`$_2$
`valorNomeServidor`$_2$ `valorCargoServidor`$_2$
...
`valorIdServidor`$_n$ `valorSalarioServidor`$_n$ `valorTelefoneServidor`$_n$
`valorNomeServidor`$_n$ `valorCargoServidor`$_n$

onde:

- `arquivo.bin` é um arquivo binário de entrada que segue as mesmas especificações definidas no primeiro 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 no primeiro trabalho prático, a saber: `idServidor`, `salarioServidor`, `telefoneServidor`, `nomeServidor`, `cargoServidor`. 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 arquivo.bin.`

Mensagem de saída caso algum erro seja encontrado:

`Falha no processamento do arquivo.`

Exemplo de execução:

```
./programaTrab2
5 arquivo.bin 2
1234 6099.57 NULO NULO "ASSISTENTE EM CIENCIA E TECNOLOGIA"
2132 NULO NULO "JOAO JOSE PEDRO DA SILVA TEIXEIRA" "ANALISTA DE SEGURO"
usar a função escreverNaTela1 ou escreverNaTela2 antes de terminar a execução da funcionalidade, para mostrar a saída do arquivo arquivo.bin, o qual foi atualizado com as inserções.
```

[6] Permita a atualização de registros. Os registros a serem atualizados devem ser aqueles que satisfaçam um critério de busca determinado pelo usuário. Por exemplo, o usuário pode solicitar a atualização de um registro que possui um determinado *identificador do servidor* ou o usuário pode solicitar a atualização de todos os registros de um determinado *cargoServidor*. Note que qualquer campo pode ser utilizado como forma de atualização. Adicionalmente, o campo utilizado como busca não precisa ser, necessariamente, o campo a ser atualizado. Por exemplo, pode-se buscar pelo campo *idServidor*, e pode-se atualizar o campo *salarioServidor*. Quando o tamanho do registro atualizado for maior do que o tamanho do registro atual, então o registro atual deve ser removido e o registro atualizado deve ser inserido como um novo registro. A implementação dessa funcionalidade deve ser realizada usando o conceito de *lista ordenada de forma crescente em termos do tamanho do registro*, e deve seguir estritamente a matéria apresentada em sala de aula. Não deve ser realizado o tratamento da fragmentação interna, sendo que o lixo que permanece no registro logicamente removido deve ser identificado pelo caractere @. Também não deve ser realizado o tratamento de fragmentação externa usando a técnica de coalescimento. Caso contrário, ou seja, quando o tamanho do registro atualizado for menor ou igual ao tamanho do registro atual, então o registro atual deve ser atualizado sem a necessidade de remoção e inserção. O lixo que permanece no registro atualizado, quando ele tiver tamanho menor, deve ser preenchido com o caractere @. Campos a serem atualizados com valores nulos devem ser identificados, na entrada da funcionalidade, com NULO. A funcionalidade [6] 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 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 escreverNaTela1 ou escreverNaTela2, ambas disponibilizadas na página do projeto da disciplina, para mostrar a saída do arquivo binário.

Entrada do programa para a funcionalidade [6]:

`6 arquivo.bin n`

$\texttt{nomeCampoBusca}_1$ $\texttt{valorCampoBusca}_1$ $\texttt{nomeCampoAtualiza}_1$ $\texttt{valorCampoAtualiza}_1$
$\texttt{nomeCampoBusca}_2$ $\texttt{valorCampoBusca}_2$ $\texttt{nomeCampoAtualiza}_2$ $\texttt{valorCampoAtualiza}_2$
...
$\texttt{nomeCampoBusca}_n$ $\texttt{valorCampoBusca}_n$ $\texttt{nomeCampoAtualiza}_n$ $\texttt{valorCampoAtualiza}_n$

onde:

- `arquivo.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 nome do campo que será utilizado para buscar o registro e o seu respectivo valor, bem como o nome do campo a ser atualizado e o seu respectivo valor. Valores nulos devem ser identificados, na entrada da funcionalidade, por NULO. O nome do campo de busca e o nome do campo a ser atualizado podem ser iguais ou diferentes. Cada uma das `n` atualizações deve ser especificada em uma linha diferente. Deve ser deixado um espaço em branco entre cada um dos parâmetros de entrada. 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 arquivo.bin.`

Mensagem de saída caso algum erro seja encontrado:

`Falha no processamento do arquivo.`

Exemplo de execução:

```
./programaTrab2
6 arquivo.bin 2
idServidor 1914208 nomeServidor "JOAO JOSE PEDRO DA SILVA TEIXEIRA"
cargoServidor "TECNICO EM SAUDE PUBLICA" cargoServidor "ENFERMEIRO"
usar a função escreverNaTela1 ou escreverNaTela2 antes de terminar a
execução da funcionalidade, para mostrar a saída do arquivo
arquivo.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 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 programaTrab2 *.c
run:
./programaTrab2
```

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 para a **Turma A**: **SRAZ**
- código de matrícula para a **Turma B**: **29QF**

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 !