Programação Estruturada (A1 e A2 diurno)

Descrição | Visualizar envios

Lab 09b - Sistema de notas

Disponível até: quinta, 29 Nov 2018, 23:55
Arquivos requeridos: lab09b.c (Baixar)
Tipo de trabalho: Trabalho individual

UFABC - CMCC
MCTA028-15 - Programação Estruturada
Laboratório 09b - Sistema de notas

Prazo de entrega: 29/11/2018

Peso: 5

Descrição

Neste laboratório vamos criar um programa para gerenciar uma base de dados com registros acadêmicos de alunos contendo as seguintes informações para cada aluno: RA, nome do aluno e telefone. As operações que poderão ser realizadas sobre a base de dados são

1. adição/edição de um aluno
2. busca de um aluno
3. remoção de um aluno
4. impressão de um registro

A base de dados será implementada com um vetor que deverá ser alocado dinamicamente.

Um registro com as informações de um aluno deverá ser implementado como um `struct` da

seguinte forma:

```
typedef struct {
    int ra, telefone;
    char nome[100];
} Aluno;
```

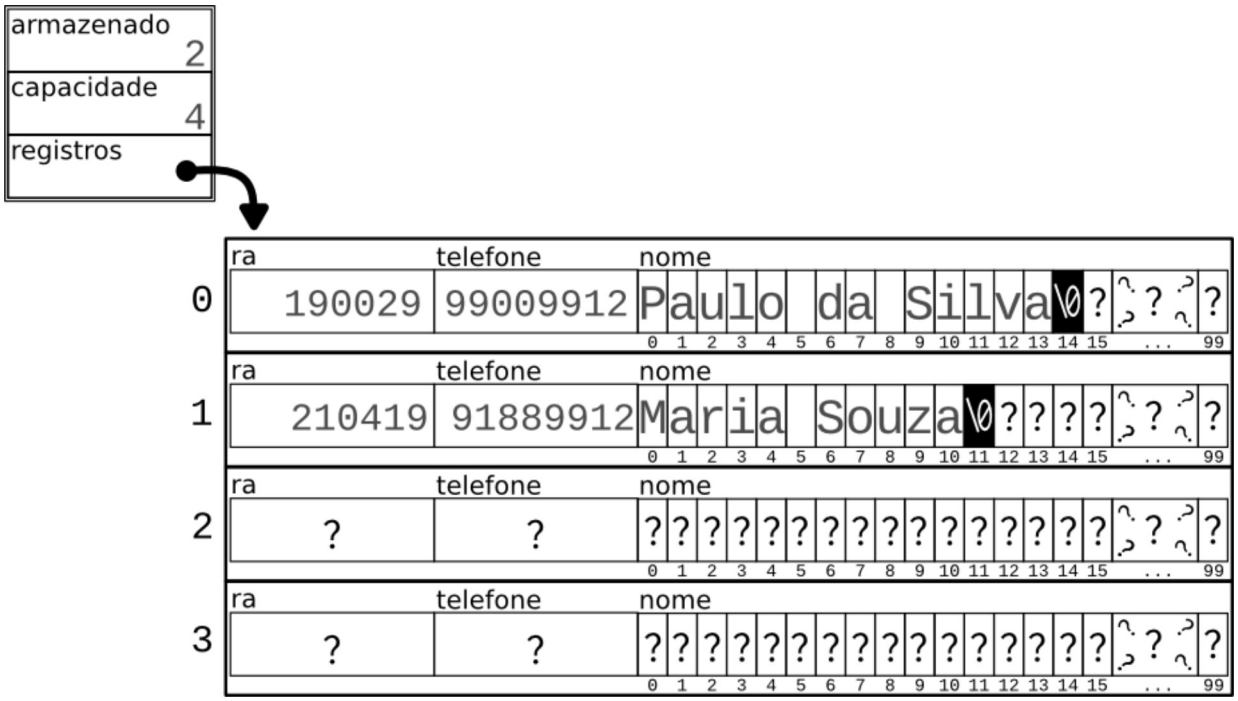
A base de dados deverá ser implementada também como um `struct` da seguinte forma:

```
typedef struct {
    int armazenados;
    int capacidade;
    Aluno *alunos;
} Base;
```

O campo `armazenados` deverá conter o número de alunos cadastrados na base, o campo `capacidade` deverá conter o tamanho alocado do vetor `alunos`, enquanto que este último deverá apontar para o vetor `alunos` alocado.

Inicialmente o programa lerá um inteiro positivo `n` que indica o número máximo de alunos na base de dados. O programa deverá então alocar o vetor `alunos` com tamanho igual a `n`.

Abaixo temos uma ilustração da nossa base de dados.



Exemplo de registros armazenados

A função main para este laboratório já foi implementada. Você deverá implementar 6 funções, que realizam as seguintes operações:

- *Criar a Base*: Esta função recebe como parâmetros um ponteiro para a base de dados e um inteiro positivo `n`. A função deve definir o campo `capacidade` para `n` e o campo `armazenado` para `0`. A função deve também alocar o vetor `alunos` com tamanho igual a `n` e imprimir a mensagem `Base criada.\n`.

- *Liberar a Base*: Esta função recebe como parâmetro um ponteiro para a base de dados e deve liberar a memória alocada e definir os valores dos campos `capacidade` e `armazenado` para zero, e `alunos` para `NULL` .
- *Buscar*: Esta função recebe como parâmetros um ponteiro para a base de dados e o RA de um aluno. A função deve verificar se o RA informado está na base, retornando o índice do vetor `alunos` que contém o aluno com o RA informado, ou -1 caso não seja encontrado nenhum aluno com este RA.
- *Adicionar/Alterar*: Esta função recebe como parâmetros um ponteiro para a base de dados, e os dados de um aluno, que são RA, nome e telefone. A função deve incluir o aluno no vetor `alunos` ou alterar o registro de mesmo RA, se o aluno com este RA já estiver presente na base. A função deverá imprimir as informações do aluno no formato `Adicionado: RA - TELEFONE - NOME\n` ou `Alterado: RA - TELEFONE - NOME\n` dependendo se o aluno tenha sido adicionado ou tenha tido seus dados alterados. A função deve também ajustar o campo `armazenado` caso um novo aluno seja incluído. Caso o vetor `alunos` esteja cheio para inclusão do aluno a função deve imprimir a mensagem `Erro: base cheia.\n` .
- *Imprimir*: Esta função recebe como parâmetros um ponteiro para a base de dados e o RA de um aluno. A função deve imprimir as informações do aluno informado no formato `RA - TELEFONE - NOME\n` . Caso o aluno não esteja cadastrado a função deve imprimir `Aluno RA nao encontrado.\n` , onde no lugar de `RA` deverá ser impresso o RA do aluno.
- *Remover*: Esta função recebe como parâmetros um ponteiro para a base de dados e o RA de um aluno. A função deve então remover o aluno com o RA informado da base, e ajustar o campo `armazenado` caso seja necessário. Caso o aluno tenha sido removido a função deve imprimir `Aluno RA removido.` , e caso o aluno não esteja na base a função deverá imprimir `Aluno RA nao encontrado.\n` , onde no lugar de `RA` deverá ser impresso o RA do aluno.

Para esse laboratório você só deverá implementar as funções descritas acima em um arquivo chamado `lab09b.c` , que está parcialmente implementado com as assinaturas dessas funções.

A função principal (`main`) está fornecida em um arquivo separado, chamado `lab09b_main.c` .

Arquivos auxiliares podem ser obtidos em http://professor.ufabc.edu.br/~e.francesquini/pe/files/testes_abertos/.

A descrição geral dos parâmetros de entrada e saída das funções encontra-se nos comentários das assinaturas das funções.

Entrada

A primeira linha da entrada contém um inteiro `n` indicando o número máximo de alunos que a base conterá.

As linhas seguintes consistem de operações a serem realizadas na base de alunos. As operações são:

- `> RA` : Imprimir o registro de um aluno com o RA informado, ou uma mensagem dizendo que o aluno não foi encontrado;
- `+ RA TELEFONE NOME` : Adicionar/Alterar um aluno com os dados fornecidos;

- `- RA` : Remover o aluno de RA informado da base, ou uma mensagem dizendo que o aluno não foi encontrado;
- `q` : Limpar a base e encerrar o programa;

onde:

- `RA` é o número do RA;
- `TELEFONE` é o número do telefone sem formatação (somente dígitos);
- `NOME` é o nome do aluno.

Saída

Cada linha da saída do programa contém o resultado da execução de cada operação dada na entrada, de forma que a saída possui uma linha a menos que a quantidade de linhas da entrada.

Exemplos

Teste 01

Entrada:

```
2
> 190029
+ 190029 99009912 Paulo da Silva
> 190029
+ 210419 91889912 Maria Souza
q
```

Saída:

```
Base criada.
Aluno 190029 nao encontrado.
Adicionado: 190029 - 99009912 - Paulo da Silva
190029 - 99009912 - Paulo da Silva
Adicionado: 210419 - 91889912 - Maria Souza
```

Teste 02

Entrada:

```
2
> 190029
+ 190029 99001111 Paulo de Silva
> 190029
+ 210419 91889912 Maria Souza
+ 190030 99009913 Victor da Silva
+ 210419 10000001 Maria Souza
> 190029
- 190029
> 190029
> 210419
- 210419
> 210419
q
```

Saída:

```
Base criada.
Aluno 190029 nao encontrado.
Adicionado: 190029 - 99001111 - Paulo de Silva
190029 - 99001111 - Paulo de Silva
Adicionado: 210419 - 91889912 - Maria Souza
Erro: base cheia.
Alterado: 210419 - 10000001 - Maria Souza
190029 - 99001111 - Paulo de Silva
Aluno 190029 removido.
Aluno 190029 nao encontrado.
210419 - 10000001 - Maria Souza
Aluno 210419 removido.
Aluno 210419 nao encontrado.
```

Para mais exemplos, consulte os testes abertos em http://professor.ufabc.edu.br/~e.francesquini/pe/files/testes_abertos/.

Observações

- O número máximo de submissões é 15;
- Para a realização dos testes, a compilação dos programas desenvolvidos em C irá considerar o comando: `gcc -ansi -pedantic -Wall -Werror -o lab09b lab09b.c` .
- Você deve incluir, no início do seu programa, uma breve descrição dos objetivos do programa, da entrada e da saída, além do seu nome e do seu RA.
- Indente corretamente o seu código e inclua comentários no decorrer do seu programa.

Critérios importantes

Independentemente dos resultados dos testes, o não cumprimento dos critérios abaixo implicará em nota zero nesta tarefa de laboratório.

- Os únicos headers aceitos para essa tarefa serão o `stdio.h` , `stdlib.h` e `string.h` .

VPL

Você acessou como Lucas Fernandes Muniz (Sair)
PE2018_DA1DA2

Português - Brasil (pt_br)
English (en)

Português - Brasil (pt_br)