Instituto de Computação - Unicamp

MC102 - Algoritmos e Programação de Computadores

Laboratório 17 - Sistema de Notas

Prazo de entrega: **09/06/2017 23:59:59**

Peso: **2**

Professor: Eduardo C. Xavier
Professor: Guido Araújo

Monitor: Arthur Pratti Dadalto
Monitor: Cristina Cavalcante
Monitor: Klairton de Lima Brito
Monitor: Luís Felipe Mattos
Monitor: Paulo Finardi
Monitor: Paulo Lucas Rodrigues Lacerda
Monitor: Pedro Alves
Monitor: Renan Vilas Novas
Monitor: Vinicius de Novaes Guimarães Pereira

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 i) adição/edição de um aluno ii) busca de um aluno iii) remoção de um aluno e iv) 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;
```

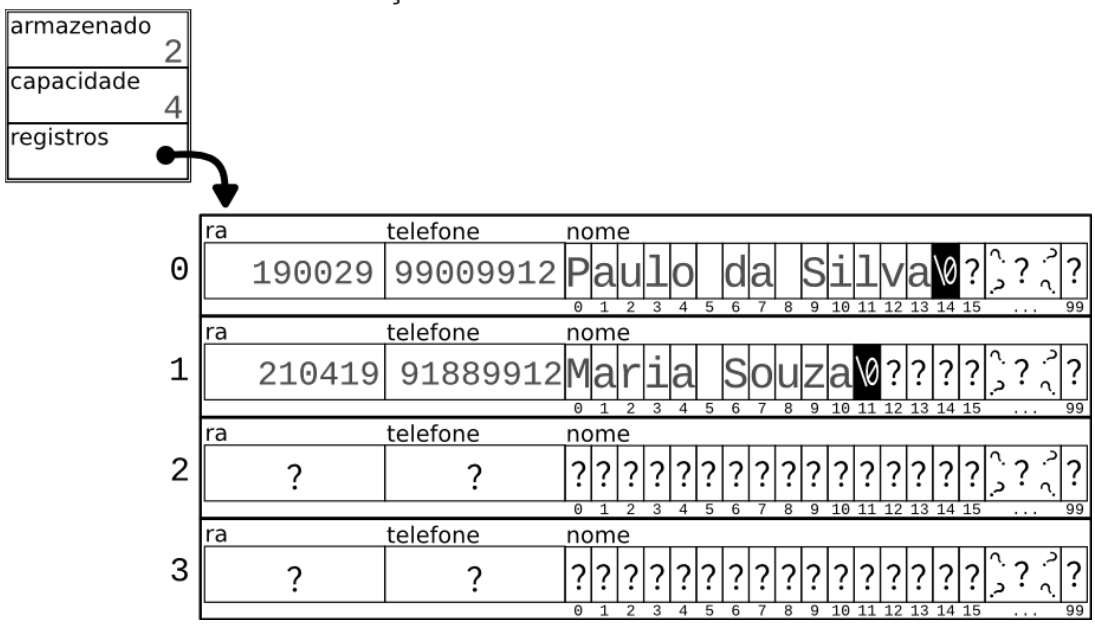
A base de dados deverá ser implementada também como um `struct` da seguinte forma:

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

O campo `armazenado` 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.



A função main para este laboratório já foi implementada. Você deverá apenas 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 `lab17.c`, que está parcialmente implementado com as assinaturas dessas funções. A função principal (`main`) será fornecida em um arquivo separado, chamado `lab17_main.c`. Links para ambos os arquivos estão disponíveis na página do laboratório. A descrição geral dos parâmetros de entrada e saída das funções encontra-se nos comentários das assinaturas das funções, que são fornecidas a seguir.

```
#include <stdio.h>
#include <stdlib.h>

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

typedef struct {
  int armazenado;
  int capacidade;
  Aluno *alunos;
} Base;

/* Funcao: criar_base
 *
 * Inicializa a base ja com a capacidade.
 *
 * Parametros:
 *   base: ponteiro para a base
 *   n: quantidade maxima de alunos
 */
void criar_base(Base *base, int n) {
  /* Implementar */
  return;
}

/* Funcao: buscar
 *
 * Parametros:
 *   base: ponteiro para a base
 *   ra: numero do RA
```

```
 *
 * Retorno:
 *   Indice do registro com RA no vetor de alunos
 *   -1 caso contrario.
 */
int buscar(Base *base, int ra) {
  /* Implementar */
  return -1;
}

/* Funcao: imprimir
 *
 * Parametros:
 *   base: ponteiro para a base
 *   ra: numero do RA
 */
void imprimir(Base *base, int ra) {
  /* Implementar */
  return;
}

/* Funcoes: adicionar
 *
 * Inclui um registro sem permitir RAs duplicados.
 * O quantidade de alunos deve ser atualizada.
 *
 * Parametros:
 *   base: ponteiro para a base
 *   ra: numero do RA
 *   telefone: numero do telefone
 *   nome: string com o nome
 */
void adicionar(Base *base, int ra, int telefone, char *nome) {
  /* Implementar */
  return;
}

/* Funcoes: remover
 *
 * Remove um registro se o ra estiver presente.
 * O quantidade de registro deve ser atualizada.
 *
 * Parametros:
 *   base: ponteiro para a base
 *   ra: numero do RA
 */
void remover(Base *base, int ra) {
```

```
  /* Implementar */
  return;
}

/* Funcao: liberar_base
 *
 * Libera a memoria de todos alunos da base.
 * Deve deixar a base com capacidade e quantidade armazenada igua
 * e o ponteiro para alunos igual a NULL.
 *
 * Parametros:
 *   base: ponteiro para a base
 */
void liberar_base(Base *base) {
  /* Implementar */
  return;
}
```

Reforçando

Neste laboratório você não precisará se preocupar em ler a entrada a partir da entrada padrão. Seu trabalho é apenas implementar as funções descritas. A função `main()` que é fornecida no arquivo `lab17_main.c` se encarrega da leitura e de chamar as funções.

Você também não deve submeter o arquivo `lab17_main.c` para o SuSy, **somente o arquivo** `lab17.c` .

As sessões abaixo, de Entrada e Saída, descrevem os formatos de entrada e saída, mas você não precisa se preocupar com eles.

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 no Susy.

Observações

- O número máximo de submissões é **10**;
- O seu programa deve estar completamente contido em um único arquivo denominado `lab17.c` ;

- Para a realização dos testes do SuSy, a compilação se dará da seguinte forma:
  `gcc -std=c99 -pedantic -Wall -o lab17 lab17.c lab17_main.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 do SuSy, 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` .