MC202GH - Estrutura de Dados - Turmas G e H

Laboratório 3 - *TAD, registros e ponteiros*

Docente: Marcelo da Silva Reis

Monitor PED: Matheus Abrantes Cerqueira

Monitores PAD: Andreas Cisi Ramos
Wallace Gustavo Santos Lima

18 de setembro de 2022

Data de entrega: 30/9/2022

Entrega no codePost[1]

Informações gerais

O presente laboratório conclui a parte prática das aulas de introdução à linguagem C, com foco na manipulação de registros e de ponteiros. Para isso, serão fornecidos arquivos com protótipos a serem modificados e enviados na plataforma de avaliação (serão três envios, um para cada questão: "Trabalho 03 - Qx", sendo x o número da questão.).

Treinaremos também neste laboratório a organização do código através de Tipo Abstrato de Dados (TAD), uma maneira de organizar o código de forma que objetos (estruturas) e operadores (funções) para sua manipulação são prototipados em um arquivo de *interface* (.h), a implementação concreta das operações ocorre em um arquivo de *implementação* (.c com mesmo nome do .h) e o uso dos objetos e operadores propriamente ditos é feito em um arquivo de *cliente* (.c que contém o main).

Observações importantes:

1. Neste laboratório será permitido o uso apenas das bibliotecas `stdio.h`, `stdlib.h`, `string.h` e `math.h`, além do arquivo de interface do laboratório, que é o `ficha.h`.

2. O arquivo `ficha.h` que contém os protótipos de tipos e operadores do nosso TAD, não deve ser modificado em hipótese alguma. Alterações devem ser feitas apenas nos arquivos de implementação e de cliente (**ficha.c** e **questao_x.c**).

[1] https://codepost.io/signup/join?code=ZW239C3IID

3. Para cada questão, entre na pasta correspondente à mesma e compile o código utilizando o arquivo `Makefile` contindo nela.

TAD para manipulação de fichas de estudantes

Suponhamos que temos que implementar um sistema simplificado de cadastro ficha de estudantes da Unicamp. Para isso, coletaremos somente o primeiro nome e o último sobrenome da pessoa, seu RA e seu CR, nessa ordem. Nosso TAD então terá como objetos fichas de estudantes (implementadas como `struct`), e os seguintes operadores para manipulá-las:

- `busca_ficha`, para devolver o índice da ficha para um determinado RA;
- `calcula_media_CR`, para devolver o CR médio de todos os estudantes cadastrados;
- `cria_fichario`, para iniciar um novo vetor de fichas;
- `imprime_ficha`, para imprimir na saída padrão uma determinada ficha;
- `insere_ficha`, para inserir no vetor de fichas uma determinada ficha;
- `le_ficha`, para ler os dados de uma ficha da entrada padrão;
- `remove_ficha`, para remover do vetor de fichas a ficha com um determinado RA.

Os operadores acima estão prototipados como funções no arquivo `ficha.h`, acompanhados de documentação com descrições sobre os tipos dos parâmetros que recebem, o que fazem e o que devolvem. Esse arquivo também contém a definição do tipo `ficha`, implementado como `struct` como mencionado acima.

Nas questões a seguir iremos implementar gradualmente, no arquivo `ficha.c`, cada um desses operadores.

Questão 1 (5 pontos) - Cliente e inserção de fichas

Nesta questão implementaremos no arquivo `questao_1.c` um loop para ler da entrada padrão uma sequência de comandos e de fichas de alunos, um comando ou uma ficha por linha. Os comandos são os seguintes:

- **N** k: cria um vetor de fichas (utilizando a função `malloc`), de tamanho especificado pelo inteiro k. Todas as fichas do vetor alocado devem ter o membro RA setado com zero;
- **I** k: insere k fichas no vetor de fichas. Essas fichas são fornecidas nas k linhas seguintes ao comando, uma ficha por linha. Se não for possível inserir uma ficha porque o vetor está cheio (uma posição do vetor está ocupada se RA é diferente de zero), então deverá ser impressa na tela a expressão "Erro inserindo " seguida do primeiro nome do estudante em questão (observe que o programa não termina com erro);
- **P**: imprime na saída padrão todas as fichas cadastradas, na sequência CR, RA, primeiro nome, último sobrenome;
- **C**: imprime na saída padrão o CR médio dos estudantes, com duas casas decimais de precisão;

- **F**: finaliza a execução do programa.

Por exemplo, se para executar o seu programa for fornecida a seguinte sequência de comandos:

```
N 3
I 2
Tio Patinhas 112233 0.9
Professor Pardal 778899 1.0
P
I 2
Pato Donald  445566 0.55
Mickey Mouse 001122 0.8
P
C
F
```

O seu programa terá que resultar na seguinte saída:

```
0.90 112233 Tio Patinhas
1.00 778899 Professor Pardal
Erro inserindo Mickey
0.90 112233 Tio Patinhas
1.00 778899 Professor Pardal
0.55 445566 Pato Donald
CR da turma: 0.82
```

Após codificar o cliente e também as operações necessárias na implementação, compile tudo utilizando o `Makefile` da questão 1 e teste executando o seguinte comando:

```
./questao_1.bin < teste_01.in
```

onde `teste.in` é um arquivo contendo uma sequência de comandos como a exemplificada acima.

Questão 2 (3 pontos) - Busca e remoção de fichas

Agora vamos expandir o cliente da questão anterior, codificando mais duas operações na implementação e acrescentando seus respectivos comandos no cliente:

- **B** $< RA >$: busca pelo registro que contém $< RA >$ no vetor de fichas. Caso o mesmo não seja encontrado deverá ser impresso "RA $< RA >$ inexistente";
- **R** $< RA >$: busca pelo registro que contém $< RA >$ no vetor de fichas e o remove (setando seu RA com zero). Se o registro for removido com sucesso deverá ser impresso "RA $< RA >$ removido"; caso contrário, deverá ser impresso "RA $< RA >$ inexistente".

Por exemplo, se para executar o seu programa for fornecida a seguinte sequência de comandos:

```
N 3
I 2
Tio Patinhas 112233 0.9
Professor Pardal 778899 1.0
P
I 2
Pato Donald  445566 0.55
Mickey Mouse 001122 0.8
P
B 778899
B 666666
R 778899
B 778899
R 111111
I 1
Mickey Mouse 001122 0.8
P
C
F
```

O seu programa terá que resultar na seguinte saída:

```
0.90 112233 Tio Patinhas
1.00 778899 Professor Pardal
Erro inserindo Mickey
0.90 112233 Tio Patinhas
1.00 778899 Professor Pardal
0.55 445566 Pato Donald
1.00 778899 Professor Pardal
RA 666666 inexistente
RA 778899 removido
RA 778899 inexistente
RA 111111 inexistente
0.90 112233 Tio Patinhas
0.80 001122 Mickey Mouse
0.55 445566 Pato Donald
CR da turma: 0.75
```

Questão 3 (2 pontos) - Vetor dinâmico de fichas

Nas questões anteriores, uma vez que o vetor de fichas esteja cheio não é possível mais inserir alunos. Podemos contornar isso utilizando o truque de tabelas dinâmicas: sempre que o vetor alocado estiver cheio, aloque um novo vetor maior (com o dobro do tamanho do vetor original), copie as fichas para o novo vetor e libere da memória (com a função `free`) o vetor velho.

Fazendo isso, se ao executar o seu terceiro programa for fornecida a seguinte sequência de comandos:

```
N 3
I 2
Tio Patinhas 112233 0.9
Professor Pardal 778899 1.0
P
I 2
Pato Donald  445566 0.55
Mickey Mouse 001122 0.8
P
B 778899
B 666666
R 778899
B 778899
R 111111
I 2
Minnie Mouse 335544 0.95
Buzz Lightyear 889977 0.6
P
C
F
```

O seu programa terá que resultar na seguinte saída:

```
0.90 112233 Tio Patinhas
1.00 778899 Professor Pardal
0.90 112233 Tio Patinhas
1.00 778899 Professor Pardal
0.55 445566 Pato Donald
0.80 001122 Mickey Mouse
1.00 778899 Professor Pardal
RA 666666 inexistente
RA 778899 removido
RA 778899 inexistente
RA 111111 inexistente
0.90 112233 Tio Patinhas
0.95 335544 Minnie Mouse
0.55 445566 Pato Donald
0.80 001122 Mickey Mouse
0.60 889977 Buzz Lightyear
CR da turma: 0.76
```

Dica: para esta questão não é necessário alterações no cliente da questão anterior; basta uma modificação no arquivo de implementação.