MC202GH - Estrutura de Dados - Turmas G e H

Laboratório 2 - *Gerenciando referências bibliográficas*

Docente: Marcelo da Silva Reis

Monitores PED: Cristiano Gabriel de Souza Campos
Heigon Alafaire Soldera Pires

Monitores PAD: Gabryel Rodrigues Alves Da Silva
Hermes Shimidu

1 de setembro de 2023

Data de entrega: 10/09/2023

Entrega no GitHub Classroom[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. Treinaremos também neste laboratório o conceito de Tipo Abstrato de Dados (TAD), uma maneira de organizar o código de forma que objetos (registros) 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 **cliente.c**).
3. Para compilar a sua questão, utilize o `Makefile`, chamando no terminal o comando `make`.

[1] classroom.github.com/a/9OvqWEne.

TAD para manipulação de fichas de referências bibliográficas

Você é um(a) estudante muito diligente, e por conta disso pretende usar suas recém-descobertas habilidades de programação em C para escrever um programa para organizar os artigos científicos que você precisa ler em diferentes disciplinas (e em sua Iniciação Científica, caso esteja fazendo uma).

Para isso, coletaremos informações sobre diferentes artigos da entrada padrão, armazendo dados de cada artigo em um registro. A estrutura do registro deverá conter os seguintes campos:

- Número DOI (Digital object identifier) do artigo;
- Número de autores;
- Últimos sobrenomes dos autores;
- Ano de publicação;
- Volume.

Para manipular esses registros, implementaremos os seguintes operadores:

- `busca_ficha`: devolve o índice da ficha para um determinado DOI;
- `cria_fichario`: inicia um novo vetor de fichas;
- `imprime_ficha`: imprime na saída padrão uma determinada ficha;
- `insere_ficha`: insere no vetor de fichas uma determinada ficha;
- `le_ficha`: lê os dados de uma ficha da entrada padrão;
- `remove_ficha`: remove do vetor de fichas a ficha com um determinado DOI;
- `destroi_ficha`: libera um vetor de fichas da memória.

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 `cliente.c` um loop para ler da entrada padrão uma sequência de comandos e de fichas de referências bibliográficas, 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 DOI setado como string vazia;
- **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 o DOI é diferente de string vazia), então deverá ser impressa na tela a expressão "Erro ao inserir DOI " seguida do DOI do artigo em questão (observe que o programa não termina com erro);
- **P**: imprime na saída padrão todas as fichas cadastradas, na sequência DOI, sobrenomes dos 3 primeiros autores e ano de publicação (se o artigo tiver mais do que 3 autores, inclua o "et al"após o sobrenome do terceiro autor);
- **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
10.48550/arXiv.1706.03762 7 Vaswani Shazeer Parmar Uszkoreit Jones Gomez Kaiser 2017 30
10.1038/s41586-021-03819-2 4 Jumper Evans Pritzel Green 2021 596
P
I 2
10.1145/130385.130401 3 Boser Guyon Vapnik 1992 5
10.1109/5.726791 4 Lecun Bottou Bengio Haffner 1998 86
P
F
```

O seu programa terá que resultar na seguinte saída:

```
10.48550/arXiv.1706.03762 Vaswani, Shazeer, Parmar, et al. (2017) 30
10.1038/s41586-021-03819-2 Jumper, Evans, Pritzel, et al. (2021) 596
Erro ao inserir DOI 10.1109/5.726791
10.48550/arXiv.1706.03762 Vaswani, Shazeer, Parmar, et al. (2017) 30
10.1038/s41586-021-03819-2 Jumper, Evans, Pritzel, et al. (2021) 596
10.1145/130385.130401 Boser, Guyon, Vapnik (1992) 5
```

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

```
./cliente.bin < teste_Q1.in
```

onde `teste_Q1.in` é um arquivo contendo uma sequência de comandos como a exemplificada acima.

Questão 2 (5 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** *DOI*: busca pelo registro que contém *DOI* no vetor de fichas e o imprime **sem o DOI**. Caso o mesmo não seja encontrado deverá ser impresso "*DOI* inexistente";
- **R** *DOI*: busca pelo registro que contém *DOI* no vetor de fichas e o remove (setando como string vazia). Se o registro for removido com sucesso deverá ser impresso "DOI *DOI* removido"; caso contrário, deverá ser impresso "DOI *DOI* inexistente".

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

```
N 3
I 2
10.48550/arXiv.1706.03762 7 Vaswani Shazeer Parmar Uszkoreit Jones Gomez Kaiser 2017 30
10.1038/s41586-021-03819-2 4 Jumper Evans Pritzel Green 2021 596
P
I 2
10.1145/130385.130401 3 Boser Guyon Vapnik 1992 5
10.1109/5.726791 4 Lecun Bottou Bengio Haffner 1998 86
P
B 10.1038/s41586-021-03819-2
B 10.0000/11111
R 10.1038/s41586-021-03819-2
B 10.1038/s41586-021-03819-2
R 10.0000/22222
I 1
10.1109/5.726791 4 Lecun Bottou Bengio Haffner 1998 86
P
F
```

O seu programa terá que resultar na seguinte saída:

```
10.48550/arXiv.1706.03762 Vaswani, Shazeer, Parmar, et al. (2017) 30
10.1038/s41586-021-03819-2 Jumper, Evans, Pritzel, et al. (2021) 596
Erro ao inserir DOI 10.1109/5.726791
10.48550/arXiv.1706.03762 Vaswani, Shazeer, Parmar, et al. (2017) 30
10.1038/s41586-021-03819-2 Jumper, Evans, Pritzel, et al. (2021) 596
10.1145/130385.130401 Boser, Guyon, Vapnik (1992) 5
Jumper Evans Pritzel et al. (2021) 596
DOI 10.0000/11111 inexistente
DOI 10.1038/s41586-021-03819-2 removido
DOI 10.1038/s41586-021-03819-2 inexistente
DOI 10.0000/22222 inexistente
10.48550/arXiv.1706.03762 Vaswani, Shazeer, Parmar, et al. (2017) 30
10.1109/5.726791 Lecun, Bottou, Bengio, et al. (1998) 86
10.1145/130385.130401 Boser, Guyon, Vapnik (1992) 5
```