BCC202 – Estruturas de Dados I (2023-01)

Departamento de Computação - Universidade Federal de Ouro Preto - MG
Professor: **Pedro Silva** (www.decom.ufop.br/)

AULA PRÁTICA 5

- Data de entrega: Até 25 de junho às 23:59:59.

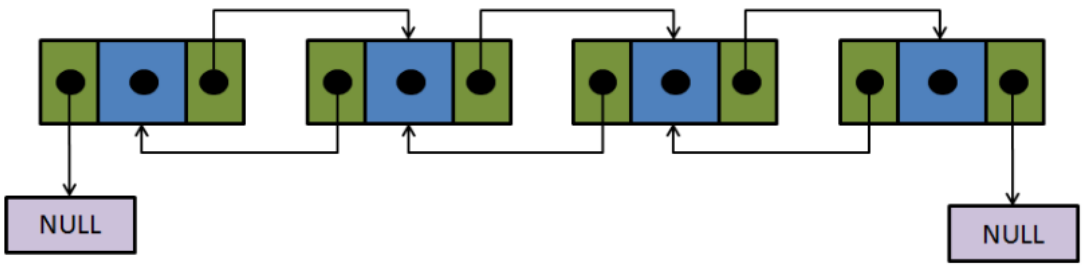
- Procedimento para a entrega:.

1. Submissão: via ***Moodle***.
2. Os nomes dos arquivos e das funções devem ser especificados considerando boas práticas de programação.
3. Funções auxiliares, complementares aquelas definidas, podem ser especificadas e implementadas, se necessário.
4. A solução deve ser devidamente modularizada e separar a especificação da implementação em arquivos *.h* e *.c* sempre que cabível.
5. Os arquivos a serem entregues, incluindo aquele que contém *main()*, devem ser compactados (*.zip*), sendo o arquivo resultante submetido via ***Moodle***.
6. Caracteres como acento, cedilha e afins não devem ser utilizados para especificar nomes de arquivos ou comentários no código.
7. Siga atentamente quanto ao formato da entrada e saída de seu programa, exemplificados no enunciado.
8. Durante a correção, os programas serão submetidos a vários casos de testes, com características variadas.
9. A avaliação considerará o tempo de execução e o percentual de respostas corretas.
10. Eventualmente, serão realizadas entrevistas sobre os estudos dirigidos para complementar a avaliação.
11. Considere que os dados serão fornecidos pela entrada padrão. Não utilize abertura de arquivos pelo seu programa. Se necessário, utilize o redirecionamento de entrada.
12. Os códigos fonte serão submetidos a uma ferramenta de detecção de plágios em software.
13. Códigos cuja autoria não seja do aluno, com alto nível de similaridade em relação a outros trabalhos, ou que não puder ser explicado, acarretará na perda da nota.
14. Códigos ou funções prontas específicos de algoritmos para solução dos problemas elencados não são aceitos.
15. Não serão considerados algoritmos parcialmente implementados.

- Bom trabalho!

Movendo itens de uma lista duplamente encadeada

Em ciência da computação, uma lista duplamente ligada (ou lista duplamente encadeada) é uma estrutura de dados ligada que consiste de um conjunto de registros sequencialmente ligados chamados de nós e é uma extensão da lista simplesmente ligada (ou lista simplesmente encadeada). Cada nó contem dois campos, chamados de links ou enlaces, que são referências para o nó anterior e para o nó posterior na sequência de nós. Os links anteriores e posteriores dos nós inicial e final, respectivamente, apontam para algum tipo de terminador, tipicamente um nó sentinela ou nulo, para facilitar o percorrimento da lista.



A sua tarefa será dada um conjunto de dados, criar uma lista duplamente encadeada e após isso receber dois

inteiros ($p1$ e $p2$) maiores ou iguais a zero e trocar os elementos dessas duas posições dentro da lista. **Essa troca deve ser feita somente alterando ponteiros e não o conteúdo dos itens**.

Considerações

O código-fonte deve conter uma `TAD Lista` para efetuar a operação desejada e deve ser modularizado corretamente conforme os arquivos de protótipo fornecidos. A informação lida da entrada deve ser armazenada em um tipo abstrato de dados criado especificamente para isso (`TAD Lista`) que armazena as informações de um `TAD Aluno`. O TAD deve ser alocada e liberada a cada teste.

- Não altere o nome dos arquivos.
- O arquivo `.zip` deve conter na sua raiz somente os arquivos-fonte.
- Há vários casos de teste. Você terá acesso (entrada e saída) de casos específicos para realizar os seus testes.

Especificação da Entrada e da saída

Seu programa deve ler um único conjunto de teste. A primeira linha de um conjunto de teste contém um inteiro não negativo, N, que indica o número de registros de alunos. Seguem-se N linhas, cada uma contendo as informações de um aluno. Cada aluno possui um nome de até 20 caracteres, coeficiente (número decimal) e matrícula (cadeia de 9 caracteres). Após as N linhas lidas, são informados dois inteiros $p1$ e $p2$ que são posições da lista. Os itens dessa lista devem ser trocados de posição, ou seja, o elemento que está em $p1$ deve ocupar a posição $p2$ e o item que estava em $p2$, deve ocupar a posição $p1$.

Seu programa deve imprimir a lista resultante mostrando a matrícula do aluno seguida do coeficiente (usar somente uma casa decimal). Cada aluno deve estar em uma linha.

Entrada	Saída
5 pedro 9.0 20.1.1111 guilherme 9.5 20.1.2222 maria 10.0 20.3333 jose 5.0 20.4444 clarice 7.0 20.5555 1 2	20.1.2222 9.5 20.1.1111 9.0 20.3333 10.0 20.4444 5.0 20.5555 7.0

Diretivas de Compilação

```
$ gcc -c lista.c -Wall
$ gcc -c pratica.c -Wall
$ gcc lista.o pratica.o -o exe
```

Avaliação de *leaks* de memória

Uma forma de avaliar se não há *leaks* de memória é usando a ferramenta valgrind. Um exemplo de uso é:

```
gcc -g -o exe *.c -Wall; valgrind --leak-check=yes -s ./exe < casoteste.in
```

Espera-se uma saída com o fim semelhante a:

```
==38409== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

Para instalar no Linux, basta usar: `sudo apt install valgrind`.