BCC202 – Estruturas de Dados I (2023-02)

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

AULA PRÁTICA 05

- Data de entrega: Até 10 de novembro às 23:55.

- 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!

A concessionária de carros

Suponha o seguinte cenário hipotético. Você foi contratado(a) para gerenciar uma concessionária de carros. A concessionária possui um número *n* de carros, onde cada um possui um número de chassi (inteiro), uma marca (até 20 caracteres), um número de lugares (inteiro), ano de fabricação (inteiro) e um funcionário responsável por ele (deve ser um ponteiro para funcionário). O funcionário possui um nome (até 20 caracteres) e uma matrícula (inteiro).

A empresa precisa que você crie duas importantes operações: (i) contar quantos carros que são mais novos que um ano específico e com um número de lugares específicos; e (ii) uma função para imprimir todos os carros que são cuidados por um funcionário específico. Detalhe importante, você deve imprimir ambas as funções recursivamente.

Considerações

O código-fonte deve conter duas funções recursivas para efetuar as operações desejadas e deve ser modularizado corretamente conforme os arquivos de protótipo fornecidos. Especialmente nesta aula prática, **não** será permitido criar outras funções ou procedimentos. Também **não** será permitido criar variáveis adicionais no *main*.

- Não altere o nome dos arquivos.
- O arquivo `.zip` deve conter na sua raiz somente os arquivos-fonte.

- Há alguns casos de teste e você terá acesso (entrada e saída) a alguns deles para realizar os seus testes.

Especificação da Entrada e da saída

Em cada caso de teste de entrada haverá um inteiro que representa o número *n* de carros que serão lidos. A próxima linha é um caractere *c* para identificar qual operação ser á executada: *C* para contar e *I* para imprimir. Se for lido o caractere *C*, devem ser lidos o ano e o número de lugares, se for lido, *I*, deve ser lido o nome do funcionário. As *n* linhas seguintes contém o número de chassi do carro, marca, número de lugares e o ano de fabricação, além do nome e matrícula do funcionário responsável. A saída é o resultado da operação.

Entrada	Saída
5 C 2015 2 125435 palio 4 2018 pedro 213 425435 gol 5 2012 carlos 212 987654 palio 5 2019 pedro 213 785452 fox 5 2021 carlos 212 813429 ferrari 2 2023 pedro 213	3

Entrada	Saída
5 I pedro 125435 palio 4 2018 pedro 213 425435 gol 5 2012 carlos 212 987654 palio 5 2019 pedro 213 785452 fox 5 2021 carlos 212 813429 ferrari 2 2023 pedro 213	ferrari [2023|2|813429] palio [2019|5|987654] palio [2018|4|125435]

Diretivas de Compilação

```
$ gcc -c carro.c -Wall
$ gcc -c pratica.c -Wall
$ gcc carro.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`.