UNIVERSIDADE FEDERAL ALFENAS (UNIFAL)
Bacharelado em Ciência da Computação

Disciplina	Método de realização	Data da entrega
DCE792 - AEDS 2	Código	29/11/2024 às 14h00

Professor
Iago Augusto de Carvalho (iago.carvalho@unifal-mg.edu.br)

Trabalho prático 3 - Ordenação de *structs*

Como estudado na disciplina, existem diversos métodos e algoritmos de ordenação. Neste trabalho, você deve implementar três deles e comparar seu desempenho

O que deve ser desenvolvido: Neste trabalho cada grupo (de dois ou três integrantes) deverá implementar três algoritmos. O primeiro será um método de ordenação simples, como estudado na aula 20. O segundo será um método de ordenação ótimo, como apresentado na aula 22. O terceiro é um método de ordenação em tempo linear, como visto na aula 24. O código deverá ser desenvolvido em linguagem de programação **C**, não sendo permitido o uso de bibliotecas externas ou bibliotecas STL do C++.

O código deverá contabilizar e exibir

- O tempo de execução (em milisegundos)
- O número de operações de comparação executadas
- O número de operações de troca executadas
- A memória total gasta pelo algoritmo

Entradas:Será dada uma única instância contendo diversos itens (*structs*). Esta entrada está disponível neste mesmo diretório e se chama *jogadores.csv*. Este é um arquivo *.csv* com cinco diferentes colunas:

- Nome
- Posição
- Naturalidade
- Posição
- Idade

Note que cada linha (com excessão da primeira) refere-se a um diferente jogador. A ordenação deverá ser realizada pelo campo **nome** da struct. Observe que deve-se construir um vetor com a struct completa e ordena-lo com base nesta única chave.

O código também deverá receber como entrada (**passando-se um argumento para a função main**) um inteiro indicando qual algoritmo será utilizado

1. para o método de ordenação simples
2. para o método de ordenação ótimo
3. para o método de ordenação em tempo linear

Saída esperada: Espera-se que o código imprima a lista de *structs* ordenada (com todos os campos da *struct* e não somente o seu nome). Logo após, ele deve imprimir quatro números em ponto flutuante:

- O tempo de execução (em milisegundos)
- O número de operações de comparação executadas
- O número de operações de troca executadas
- A memória total gasta pelo algoritmo

A saída deverá, necessariamente, ser formatada em quatro linhas, seguindo a ordem apresentada acima

Código base: Este terceiro trabalho não possui um código-base. É de responsabilidade de vocês se organizarem e criarem tudo a partir do zero.

Makefile: O trabalho deverá, **obrigatoriamente**, compilar com um arquivo *makefile*, sendo que um arquivo de exemplo é disponibilizado junto do código-base deste trabalho. Este *makefile* pode ser modificado caso o grupo julgue necessário.

O código deve compilar em um ambiente Linux padrão, como o disponível nos laboratórios da UNIFAL. O código deve compilar com o comando

```
make all
```

Observa-se que o trabalho que não compilar com o *makefile* disponibilizado levará nota **zero**.

Entrega e avaliação: Cada grupo deverá enviar um arquivo *.pdf* (relatório de implementação) e um arquivo *.zip* (implementação do código em linguagem C, além do *makefile*). Os arquivos deverão ser entregues no Moodle da disciplina até as 14h00 do dia 29/11/2024. Entregas em atraso não serão toleradas.

A nota do trabalho será um valor z entre 0 e 100, sendo

$$z = 100\ c\ d$$

onde c é um binário que indica se o código compila ou não utilizando o *makefile* enviado e d é um valor real entre 0 e 1 que representa a qualidade do código de acordo com os parâmetros abaixo definidos.

Método de avaliação: O relatório em formato .pdf corresponderá a 50% da nota total. De forma complementar, o código corresponderá aos 50% restantes da nota total.
No documento *.pdf* com a descrição do problema, do algoritmo e os resultados, serão avaliados:

- Uso correto da língua portuguesa
- Qualidade e clareza na apresentação das estruturas de dados
- Qualidade e clareza na apresentação do algoritmo
- Análise correta das complexidades dos algoritmos

No código serão avaliados:

- A qualidade e clareza do código
- Comentários explicativos
- Execução correta dos algoritmos
- Saída correta de acordo com a proposta
- Facilidade de uso do Makefile