Aula 04 – Exercícios

Parte 1 – Escrita: teste de mesa (entregue via Formulário no Google Classroom)

Nos slides da Aula 04, foram deixados 10 exercícios para serem executados no papel com o algoritmo estudado: Quicksort. Em seu caderno, simule a execução dos exercícios correspondentes. Ao final, você deve escanear ou fotografar as páginas com a câmera do celular, e submetê-las no formulário que acompanha esta atividade. As páginas devem ser reunidas em um único arquivo PDF e devem estar legíveis. Atenção para enviar seu arquivo no Formulário disponível na atividade, e não na seção de upload do Classroom. Lembre-se de colocar nome nas suas folhas.

Parte 2 – Codificação (entregue via Run Codes)

Exercício 1: escreva, em linguagem C pura (sem nenhum comando próprio de C++), um programa que leia o nome de um arquivo. Cada linha desse arquivo contém: o nome completo de um aluno, a sigla de seu curso e o seu número de matrícula, estando os campos separados entre si por um TAB (`\t`), como segue:

```
Joe·Rory·Andersen——→EBP→2021001
Cael·Bernice·Guzman→MBA→2021002
Ricky·Jane·Hughes——→EAM→2021003
Jose·Nicolas·Villa——→EPR→2021004
Joel·Lee·Clay——→EHD→2021005
Elian·Levi·Archer——→FBA→2021006
Lance·Brandt·Spencer——→MLI→2021007
Tobias·Randall·Knapp——→FBA→2021008
Muhammad·Abe·Olsen→FLI→2021009
Payton·Beck·Friedman——→FLI→2021010
Lennon·Ellory·Bradley——→ECO→2021011
Cyrus·June·Dunlap——→CCO→2021012
Emmett·Aryn·Chen——→FBA→2021013
```

(*as setas laranja indicam as tabulações*)

Os dados do arquivo devem ser carregados em um vetor de `struct`, representada de acordo com o seguinte modelo:

```c
typedef struct aluno
{
    char nome[100];
    char curso[4];
    int matricula;
}aluno;
```

Após carregar todo o conteúdo do arquivo para dentro de seu programa, execute o algoritmo Quicksort para ordenar todos os dados pelo nome do aluno, em ordem alfabética crescente. Ao final do processo de ordenação, mostre os dados ordenados. Cada linha a ser impressa na tela deve conter o nome completo do aluno, a sigla de seu curso e o seu número de matrícula, sendo esses campos separados por um único espaço.

Seu programa deve ser dividido em funções. É obrigatório que você escreva, pelo menos, as seguintes funções, seguindo os protótipos dados:

```
/* esta função imprime os dados dos n alunos armazenados no vetor v, sendo um aluno por linha. Recebe como parâmetros o vetor v e o tamanho n do vetor. */
void imprimeVetor(aluno *v, int n);

/* esta função abre um arquivo para leitura e carrega o seu conteúdo no vetor v. Recebe como parâmetros o vetor v e o nome do arquivo a ser lido. */
void leVetor(aluno *v, char *nomeArquivo);

/* função partição do algoritmo Quicksort. Recebe como parâmetros o vetor v a ser ordenado, o primeiro índice válido do vetor (p) e o último índice válido do vetor (r). */
int particao(aluno *v, int p, int r);

/* função que ordena o vetor usando o algoritmo Quicksort. Recebe como parâmetros o vetor v a ser ordenado, o primeiro índice válido do vetor (p) e o último índice válido do vetor (r). */
void quickSort(aluno *v, int p, int r);
```

Exercício 2: uma das partes mais críticas do algoritmo Quicksort é a implementação da função `particao()`. A implementação apresentada na Aula 04 não é aquela originalmente proposta pelo criador do Quicksort, *Sir* Tony Hoare. Ela foi proposta por um pesquisador chamado Robert Sedgewick. Entretanto, existem implementações alternativas da função `particao()`, como a versão de Lomuto, a versão original de Hoare, além de outras (D. Gries, por exemplo). Dessa forma, sua tarefa é escrever, em linguagem C pura (sem nenhum comando próprio de C++), um programa que leia um valor N. Após criar um vetor de N posições, repasse-o, juntamente com o valor de N a uma função que leia e armazene valores no vetor. Por fim, execute o algoritmo Quicksort três vezes, sendo que na primeira execução ele invoca o particionamento de Hoare; na segunda execução ele invoca o particionamento de Lomuto; e, na última execução, ele invoca o particionamento de Sedgewick. Como resposta, seu programa deve mostrar 3 linhas de texto, onde cada linha contém dois valores separados por um único espaço. O primeiro valor é a quantidade de comparações e o segundo valor é a quantidade de trocas realizadas. Na primeira linha, apresente os valores obtidos pelo método original de Hoare. Na segunda linha, os valores obtidos pelo método de Lomuto. Na terceira linha, os valores computados pelo método de Sedgewick.
Ambas as implementações (Hoare e Lomuto) podem ser encontradas, por exemplo, na seguinte referência: CORMEN, T. H. et al. *Algoritmos*. 3ª ed., 2012.

Observação: este exercício está dividido em uma parte escrita e uma parte prática de codificação. A entrega só será válida se as duas partes forem entregues.