AULA 12 – TIPOS ABSTRATOS DE DADOS

Pretende-se desenvolver o tipo abstrato de dados **MINHEAP**, que permitirá representar e operar com um **heap binário baseado num array**.

O tipo **MINHEAP** pode ser usado, através de um algoritmo simples, para identificar os **k maiores valores** de um ficheiro de valores inteiros.

Com o auxílio de um tipo de dados adicional, **FILEREADER**, é também possível usar um **MINHEAP** para **listar** ordenadamente o **conteúdo de ficheiros de texto**, efetuando a sua leitura linha-a-linha e sem os fundir dois-a-dois.

- Comece por analisar o conteúdo dos ficheiros disponibilizados e as funcionalidades definidas para cada tipo de dados.

- **MINHEAP** – Analise, complete e teste o tipo de dados.

- **TOPKVALUES** – Analise, complete e teste o exemplo de aplicação.

- **FILEMERGE** – Analise, complete e teste o exemplo de aplicação.

MIN-HEAP

- Complete a implementação de um heap binário em **MinHeap.c**.

- Teste essa implementação executando o programa **MinHeapTest** com diversos **argumentos**.
  - Argumentos numéricos são inseridos no heap.
  - O argumento **-** retira um item do heap (o menor).
  - O argumento **?** mostra uma representação do heap em forma de árvore e em forma de array.

TOPKVALUES

Considere que tem **um ficheiro** com **números inteiros** e desconhece se estes se encontram ou não ordenados. Pretende obter os **k maiores valores armazenados no ficheiro**, mas **não** quer **ordenar** os elementos do ficheiro, pois iria gastar tempo e espaço de memória a fazê-lo.

O programa **TopKValues** recebe nos **argumentos** o **valor de k** (para se obter os k maiores valores) e o **nome do ficheiro** de valores inteiros a processar.

Analise o código fornecido e os comentários, e complete o programa **TopKValues.c**.

Para testar, pode **executar** TopKValues para os vários **ficheiros *.txt** que contêm valores inteiros (aleatórios, ordenados ou na ordem inversa).

FILE MERGE

Considere que tem **muitos ficheiros** com **linhas já ordenadas** lexicograficamente e pretende **percorrer todas essas linhas por ordem crescente**, mas **não** pode **fundir os ficheiros** dois-a-dois porque isso gastaria muito espaço em disco com ficheiros auxiliares. (Imagine que os ficheiros são grandes ou que o disco só permite leitura.)

NOME: **Nº MEC:**

O programa **FileMerge** recebe nos **argumentos** vários **ficheiros pré-ordenados** e faz uma fusão das suas linhas por ordem sem precisar de criar ficheiros auxiliares.

O algoritmo baseia-se em usar um heap para manter em memória uma linha de cada um dos ficheiros de cada vez. Na verdade, além das linhas é necessário manter os ficheiros correspondentes. Para isso usa um **novo TAD** chamado **FileReader**.

Analise o código fornecido e os comentários, e complete o programa **FileMerge.c**.

Para testar, pode **executar** FileMerge com vários dos **ficheiros fileXX.txt** que contêm linhas de texto já ordenadas.