Ficha 4

Programação Imperativa

Strings e arrays ordenados

1 Funções sobre *strings*

`https://codeboard.io/projects/225893`

1. Defina uma função `int contaVogais (char *s)` que conta quantas vogais uma *string* tem.
2. Defina uma função `int retiraVogaisRep (char *s)` que remove de uma *string* todas as repetições consecutivas de vogais. A função deverá retornar o número de vogas removidas. Por exemplo, se a string `a == "Estaa e umaa string coom duuuplicadoos"`, depois de invocarmos `retiraVogaisRep a`, a string `a` deverá ter o valor `"Esta e uma string com duplicados"`.
   - Para evitar fazer muitos deslocamentos de caracteres, apresente uma definição que usa um array auxiliar onde a string resultante será construída. No final terá que copiar essa string de volta para o array argumento.
   - Altere a função que definiu acima de forma a não precisar de usar o array auxiliar.
3. Defina uma função `int duplicaVogais (char *s)` que duplica todas as vogais de uma *string*. A função deve retornar o número de caracteres acrescentados. Assuma que o array recebido como argumento tem capacidade para armazenar o resultado pretendido.
   - Mais uma vez, e de forma a evitar muitos deslocamentos de caracteres, defina esta função usando um array auxiliar para construir a string resultante.
   - Apresente ainda uma definição alternativa onde não seja de facto necessário usar o dito array auxiliar.

2 Arrays ordenados

`https://codeboard.io/projects/225895`

1. Defina uma função `int ordenado (int v[], int N)` que testa se um array de inteiros está ordenado por ordem crescente.
2. Defina uma função `void merge (int a[], int na, int b[], int nb, int r[])` que recebe dois arrays ordenados `a` e `b` (com tamanhos `na` e `nb` respectivamente) e os funde num só array ordenado `r`. Assuma que o array `r` tem capacidade para armazenar os `na+nb` elementos.
3. Defina uma função `int partition (int v[], int N, int x)` que, dado um array `v` de tamanho `N` e um inteiro `x`, reorganiza o array de forma a que começam por aparecer todos os elementos menores ou iguais a `x` seguidos dos restantes elementos. A função retorna o número de elementos que ficaram na primeira parte do array (i.e., que são menores ou iguais a x)
   - Comece por definir uma versão desta funçõa que usa um array auxiliar para construir o resultado pretendido. No final deve copiar o conteúdo desse array de volta para o array recebido como argumento.
   - Apresente uma definição alternativa para esta função que não usa um array auxiliar. Para isso considere a seguinte estratégia:

Sejam `a` e `b` dois índices do array em que se verifica a seguinte propriedade:

- todos os elementos nos índices `[0..a[` são menores ou iguais a `x`
- todos os elementos nos índices `[a..b[` são maiores do que `x`