Ficha | Arrays e Strings

> Utilize o *template* `CProgram_Template` disponível no *GitHub*:
>
> https://github.com/estsetubal-atad/CProgram_Template
>
> **How-To Repositórios**: https://www.youtube.com/watch?v=THsizwp30r0

Arrays

Considere o seguinte programa:

```c
#include <stdio.h>
#include <stdbool.h>

/* protótipos de funções... */

int main() {
  int values[] = {0, 1, 2, 7, 10, 14, 17};

  /* Invocação de funções... */

  return 0;
}

/* Implementação de funções... */
```

Implemente cada uma das seguintes funções e teste-as.

1. Implemente a função `int countEven(int arr[], int arrLength)` que retorna a contagem de elementos pares no array `arr`. Teste a função no `main` sobre o array `v`.
2. Implemente a função `int sumArray(int arr[], int arrLength)` que retorna a soma dos elementos no array `arr`. Teste a função no `main` sobre o array `v`.
3. Implemente a função `bool contains(int val, int arr[], int arrLength)` que verifica se `val` existe em `arr`. Teste a função no `main` sobre o array `v`.

Strings

Considere o seguinte programa:

```c
#include <stdio.h>
#include <stdbool.h>
#include <ctype.h>

/* protótipos de funções... */

int main() {
```

```c
  char str[100] = "eu adoro a linguagem c";

  /* Invocação de funções... */

  return 0;
}

/* Implementação de funções... */
```

Implemente cada uma das seguintes funções e teste-as.

1. Crie uma função `char charAt(char str[], int pos)` que devolve o carácter que se encontra no índice `pos`.
   - Se `pos` não for válido devolve o carácter com código ASCII `0`.
2. Crie uma função `int length(char str[])` que recebe uma string e devolve o seu tamanho.
3. Crie uma função `int countOccurences(char str[], char c)` que calcula quantas vezes ocorre `c` em `str`.
   - Nota: utilize na implementação a função anterior `int length(char*)`.
4. Crie uma função `bool onlyLetters(char str[])` que verifica se todos os caracteres na string `str` são letras.
   - Versão A: utilizando `<ctype.h>`;
   - Versão B: utilizando códigos numéricos, i.e., tabela ASCII
5. Crie uma função `void toUppercase(char str[])` que passa todos os caracteres de `str` para maísculas.
6. Crie uma função `bool equal(char str1[], char str2[])` que verifica se duas strings passadas como argumento são iguais (case-sensitive).
7. Crie uma função `bool isPalindrome(char str[])` que verifica se a string é um palíndromo (palavra ou frase que tem a mesma sequência de letras em qualquer ordem de leitura).
   - e.g., “ovo”, “ana”, “sopapos” e “Sator arepo tenet opera rotas”.
   - Implemente uma versão que acuse, e.g., “Ana” e “aNA” como palíndromos.
8. Crie uma função `bool hasDuplicates(char str[])` que verifica se existe algum caracter duplicado em `str`.
   - A função deverá ignorar *espaços* na verificação.
9. Crie uma função `void printPairs(char str[])` que imprime todos os pares possíveis de caracteres consecutivos.
   - e.g., “abcde” -> “ab”, “ac”, “ad”, “ae”, “bc”, “bd”, “be”, “cd”, “ce”, “de”