Ficha | Recursividade

> Utilize o *template* `CProgram_Template` disponível no *GitHub*:
>
> https://github.com/estsetubal-atad/CProgram_Template

Utilize o seguinte `main.c` inicial:

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

/* protótipos de funções */

int main() {
  int values[] = {19, 30, 4, 49, 21, 32, 12, 37, 8, 28};

  /* Invocação de funções... */

  return EXIT_SUCCESS;
}

/* Implementação de funções... */
```

De acordo com os algoritmos apresentados nos slides.

1. Implemente uma função que retorna a soma de todos os inteiros de um array maiores que o valor `threshold`:

```c
int sumGreaterThan(int arr[], int arrLength, int threshold)
```

- Usando uma abordagem **iterativa**
- Usando uma abordagem **recursiva**

2. Implemente uma função que retorna o maior elemento de um array de inteiros (considere que só contém números inteiros).

```c
int largestElement(int arr[], int arrLength);
```

- Usando uma abordagem **iterativa**
- Usando uma abordagem **recursiva**

Será mais fácil se criar uma função auxiliar `int max(int a, int b)`.

3. Implemente uma função que recebe um número natural `number` e retorna o seu número de dígitos.

```c
int manyDigits(unsigned int number)
```

- Usando uma abordagem **iterativa**
- Usando uma abordagem **recursiva**

Nota: Um número `number < 10` terá garantidamente 1 dígito; necessitamos de fazer a

divisão inteira por 10 sucessivamente (para “eliminar” dígitos menos significativos)

4. Considere o algoritmo recursivo *fibonacci* desenvolvido previamente.

Pesquise um **algoritmo iterativo** para a função *fibonacci* e implemente-o.

- Mais fácil ou díficil de exprimir a solução?
- Qual a complexidade algorítmica dessa solução?
- Como avalia o balanço entre a dificuldade de implementação *versus* a eficiência algorítmica? Vale a pena?