Ficha | Gestão de Memória Dinâmica

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

Memória Heap + Valgrind

1. Inicie o ficheiro `main` com o seguinte código:

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

void arrayPrint(int arr[], int arrLength);

int main() {
    srand(time(NULL)); //inicializador de numeros aleatorios
    int *arr = NULL;

    //...
    return EXIT_SUCCESS;
}

void arrayPrint(int arr[], int arrLength) {
    if(arr == NULL) {
        printf("(NULL array)\n");
        return;
    }
    printf("Array (size = %d} contents: { ", arrLength);
    for(int i=0; i<arrLength; i++) {
        printf("%d ", arr[i]);
    }
    printf("}\n");
}
```

2. Implemente/siga os seguintes passos:
    i. No `main` solicite ao utilizador um valor inteiro `len` e faça a alocação dinâmica de um array de inteiros `arr` de tamanho `len` – utilize a função `malloc` e verifique a correta alocação de memória. Caso não seja bem sucedida, termine o programa com o erro `EXIT_FAILURE`;
    ii. Imprima o conteúdo de `arr`, mas não liberte a memória;
    iii. Compile e execute através do *valgrind* e analise o *output*;
    iv. Altere o programa e liberte a memória; repita (iii).

v. Altere o programa para a utilização da função `calloc`; repita (iii).

No final o *valgrind* não deverá acusar nenhum warning/error. Faça as verificações necessárias.

3. No seguimento do código anterior, implemente os seguintes passos:

   i. Implemente a função `void arrayFillRandom(int arr[], int arrLength)` que inicializa todas as posições do array com valores aleatórios (cada valor aleatório deverá ser gerado com `int r = rand() % 51);` teste no array existente e imprima o seu conteúdo.

   ii. Reloque o array `arr` para o dobro do seu tamanho. Em caso de sucesso, atualize também `len`. Utilique a função `realloc`.

   iii. Imprima o conteúdo do array realocado.

   iv. Verifique com o *valgrind* que não existem *memory leaks*.

4. Crie uma função `int* arrayFilterEven(int arr[], int arrLength, int *filterLength)` que devolve um array alocado dinamicamente e que contém todos os valores pares presentes em `arr`; note que o tamanho do array alocado deve ser devolvido por referência através de `*filterLength`.

   - Três alternativas:
     - A) Contabiliza primeiro a quantidade de números pares para alocar o array, ou;
     - B) Realoca continuamente um array consoante as necessidades, ou;
     - C) Aloca inicialmente com tamanho `arrLength` e depois realoca para o realmente necessário.

5. Teste a função e valide a gestão de memória com o *valgrind*.