Ficha 1.16

1. Crie um módulo chamado **array** constituído pelas funções com os seguintes protótipos:

- ```
  /*
  DESCRIÇÃO: Aloca n espaços consecutivos em memória do tipo type e devolve
  um apontador para o primeiro
  PRÉ: n > 0 && existe memória livre
  PÓS: array != 0
  EXCEÇÃO: aborta caso não haja memória suficiente
  */
  array_t * initA(int n);
  ```

- ```
  /*
  DESCRIÇÃO: Liberta o espaço em memória apontado por array
  PRÉ: array != 0
  PÓS: a memória anteriormente apontada por array fica live
  */
  void freeA(array_t * array);
  ```

- ```
  /*
  DESCRIÇÃO: Preenche o array com elementos solicitados ao utilizador
  PRÉ: array != 0 && 0 < n < size(array)
  PÓS: array vai conter os valores introduzidos pelo utilizador
  */
  void fillUserA(array_t * array, int n);
  ```

- ```
  /*
  DESCRIÇÃO: Mostra os elementos em array
  PRÉ: array != 0 && 0 < n < size(array)
  */
  void printA(array_t * array, int n);
  ```

- ```
  /*
  DESCRIÇÃO: Devolve um novo array com a soma de array1 com array2
  PRÉ: array1 != 0 && array2 != 0 && size(array1) == size(array2) == n
  PÓS: arraySoma != 0
  */
  array_t * somaA(array_t * array1, array_t * array2, int n);
  ```
  ```

2. O programa deverá funcionar se utilizado o seguinte código:

```
#include <stdio.h>
#include "array.h"

int main(){
     int n;
     array_t * arrayA = 0;
     array_t * arrayB = 0;
     array_t * arrayC = 0;

     do{
          fprintf(stdout, "Quanto elementos: ");
          fscanf(stdin, "%d", &n);
     } while (n < 1);

     arrayA = initA(n);
     arrayB = initA(n);

     fprintf(stdout, "Introduza os elementos, p.f.\n");
     fillUserA(arrayA, n);

     fprintf(stdout, "Introduza os elementos, p.f.\n");
     fillUserA(arrayB, n);

     arrayC = somaA(arrayA, arrayB, n);

     fprintf(stdout, "Resultado:\n");
     printA(arrayC, n);

     freeA(arrayA);
     freeA(arrayB);
     freeA(arrayC);

     return 0;
}
```

3. Que problema a nível de alocação de memória poderá ter o programa anterior?