UNIVERSIDADE DA BEIRA INTERIOR

Programação – LEI **1º Semestre**

Frequência 2 **Resolução possível** **16/12/2024**

1.

Considere as seguintes declarações de variáveis:

```
int  **V, *W, *X;
```

(esquema de um bloco de memória)

	...	
100500	110172	**W**
	...	
100700	110160	**X**
	...	
110160	50	
110164	10	
110168	15	
110172	60	
110176	70	
110180	80	
	...	
110500	100500	**V**
	...	

Considere que **sizeof(int) = 4** e **sizeof(int *) = 8**.

Usando os valores que constam no esquema ao lado, indique, justificando com os cálculos efetuados, os valores de cada uma das seguintes expressões:

a) W + 4 = 110172 + 4 x sizeof(int) = 110172 + 4 x 4 = **110188**

b) V[0] = **110172**

c) *X + 4 = 50 + 4 = **54**

d) *(X + 2) = *(110160 + 2 x sizeof(int)) = *(110160 + 2 x 4) = *(110168) = **15**

e) V + 2 = 100500 + 2 x sizeof(int *) = 100500 + 2 x 8 = **100516**

f) *V - 2 = 110172 - 2 x sizeof(int) = 110172 - 2 x 4 = **110164**

g) **V + 2 = 60 + 2 = **62**

NOTA: se não existir resposta, indicar com ERRO

2.

Implementar um **subprograma/função** em C que dados um array 1D **A** com **N** números inteiros (A e N são parâmetros do subprograma), **determine e devolva** como resultados a **quantidade** de números **negativos** (**< 0**) e a **quantidade** de números **positivos** (**≥ 0**) do array **A**.

Versão 1: saida da quantidade de positivos por return e a quantidade de negativos como parâmetro

```
int quantidadeNumeros1 (int A[], int N, int *neg)
{
   int  k, pos;
   pos = 0;
   *neg = 0;
   for (k = 0; k <= N-1; k++)
   {
      if (A[k] >= 0)
         pos = pos + 1;
      else
         *neg = *neg + 1;
   }
   return pos;
}
```

Versão 2: saida da quantidade de positivos e de negativos como parâmetros

```
void quantidadeNumeros2 (int A[], int N, int *neg, int *pos)
{
   int  k;
   *pos = 0;
   *neg = 0;
   for (k = 0; k <= N-1; k++)
   {
      if (A[k] >= 0)
         *pos = *pos + 1;
      else
         *neg = *neg + 1;
   }
}
```

3.

Considere um ficheiro de texto de nome "**dados.txt**", em que a **primeira linha** contém **um** número **inteiro** (**N**) e as **restantes linhas** contêm **N números reais**. Implemente um **programa em C** que:

- **construa** um **array 1D X** com os **N números reais** contidos no ficheiro de texto "**dados.txt**", usando **gestão dinâmica de memória**, e
- **guarde** todos os **números positivos do array X** no ficheiro de texto "**positivos.txt**".

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   float  *X;
   int  N, k;
   FILE  *f, *g;
   f = fopen("dados.txt", "r");
   fscanf(f, "%d", &N);
   X = (float *) malloc(N * sizeof(float));
   for(k = 0; k <= N-1; k++)
   {
      fscanf(f, "%f", &X[k]);
   }
   fclose(f);
   g = fopen("positivos.txt", "w");
   for(k = 0; k <= N-1; k++)
   {
      if(X[k] >= 0)
         fprintf(g, "%f\n", X[k]);
   }
   fclose(g);
}
```

4.

Implementar uma **função recursiva** em C que dados um **array 1D A** com **N** (N ≥ 0) números **inteiros** (com A e N parâmetros da função), **determine e devolva** a **soma** dos números **positivos** de **A**.

```
int  somaPositivos (int *A, int N)
{
   int soma;
   // parte basica/terminal
   if (N == 0)
      return  0;   //  num array vazio, a soma dos seus elementos positivos é nula
   // parte geral
   soma = somaPositivos(A, N-1);
   if (A[N-1] >= 0)
      return  soma + A[N-1];
   else
      return  soma;
}
```

5.

Implementar um **subprograma** em C que dados um **array 1D X** com **N** números **reais** (X e N são parâmetros do subprograma), **remova** do **array X** todos **os números negativos, percorrendo o array apenas uma vez** (passar uma única vez por cada elemento do array).

```
void removerNegativos (float *A, int *N)
{
   int  k;
   k = 0;
   while (k < *N)
   {
      if (A[k] < 0)
      {
         A[k] = A[*N-1];
         *N = *N - 1;
      }
      else
         k = k + 1;
   }
}
```