UNIVERSIDADE DA BEIRA INTERIOR

Programação – LEI + LMA **1º Semestre**

Exame Época Recurso **Resolução (sugestão)** **2022/2023**

1.

Escreva uma **instrução de atribuiçao** em linguagem C para cada uma das seguintes acções:

(a) atribua à variável **B** o dobro a parte inteira do valor da variável do tipo real **X**

(b) atribua à variável **P** o valor **5** se o valor da variável do tipo inteiro **N** for par e **-5** se **N** for ímpar

(c) atribuir à variável **SOMA** a soma dos algarismos das dezenas e das unidades do valor da variável do tipo inteiro **A** (Ex: A = 3572 ==> SOMA = 9 (7+2))

(a) B = 2 * (int) X;

(b) P = 5 - (N % 2) * 10;

(c) SOMA = A / 10 % 10 + A % 10;

Supondo que **X = -10**, **Y = 10** e **Z = 5**, **indique** a **ordem de cálculo dos operadores** e **determine** o **valor** de **cada** uma das seguintes **expressões**. Justifique, apresentando os cálculos efetuados.

	1		**2**		**3**		**4**		**5**		**6**	
(d)	Y	**<**	Z	*****	(3	**+**	X)	**&&**	X	**>=**	-2	*****	Z
(e)	(Z	**+**	X)	*****	10.0	*****	(11	**/**	3	**/**	3)		
(f)	Z	**+**	(2	**+**	Y)	*****	4	**/**	(Z	**−**	X)		

(d) ordem de cálculo: **3, 2, 6, 1, 5, 4**

resultado: **0 (Falso)**

```
10 < 5 * ( 3 + -10 ) && -10 >= -2 * 5    (aplicar operador 3)
10 < 5 * -7 && -10 >= -2 * 5             (aplicar operador 2)
10 < -35 && -10 >= -2 * 5                (aplicar operador 6)
10 < -35 && -10 >= -10                   (aplicar operador 1)
0(F) && -10 >= -10                       (aplicar operador 5)
0(F) && 1(V)                             (aplicar operador 4)
0(F)
```

(e) ordem de cálculo (aplicação dos operadores): **1, 4, 5, 2, 3**

resultado: **-50.0**

```
( 5 + -10 ) * 10.0 * ( 11 / 3 / 3 )     (aplicar operador 1)
-5 * 10.0 * ( 11 / 3 / 3 )               (aplicar operador 4)
-5 * 10.0 * ( 3 / 3 )                    (aplicar operador 5)
-5 * 10.0 * 1                            (aplicar operador 2)
-50.0 * 1                                (aplicar operador 3)
-50.0
```

(f) ordem de cálculo (aplicação dos operadores): **2, 5, 3, 4, 1**

resultado: **8**

```
5 + ( 2 + 10 ) * 4 / ( 5 − -10 )    (aplicar operador 2)
5 + 12 * 4 / ( 5 − -10 )            (aplicar operador 5)
5 + 12 * 4 / 15                     (aplicar operador 3)
5 + 48 / 15                         (aplicar operador 4)
5 + 3                               (aplicar operador 1)
8
```

2.

Construa um algoritmo, usando um fluxograma, que:
- peça ao utilizador que **insira/leia** vários números **inteiros** (**termina** quando inserir o **zero** (0));
- determine a **quantidade** de números **positivos** inseridos e a **soma** dos números **positivos** inseridos;
- mostre os **resultados** obtidos antes (quantidade e soma).

NOTA: Não usar arrays

3.

Construa um **programa em C** que:

- peça ao utilizador que **insira/leia** vários números **inteiros** (**termina** quando inserir o **zero** (0));
- determine a **quantidade** de números **positivos** inseridos e a **soma** dos números **positivos** inseridos;
- mostre os **resultados** obtidos antes (quantidade e soma).

NOTA: Não usar arrays

```
#include <stdio.h>   // Versão 1: usando while(){ … }
main(){
  int  A, soma, cont;
  soma = 0;
  cont = 0;
  printf("Insira um inteiro (0 para terminar): ");
  scanf("%d", &A);
  while (A != 0){
     if (A > 0){
        soma = soma + A;
        cont = cont + 1:
     }
     printf("Insira um inteiro (0 para terminar): ");
     scanf("%d", &A);
  }
  printf("Soma = %d  e  quantidade = %d\n", soma, cont);
}

#include <stdio.h>   // Versão 2: usando do{ … }while
main(){
  int  A, soma, cont;
  soma = 0;
  cont = 0;
  do{
     printf("Insira um inteiro (0 para terminar): ");
     scanf("%d", &A);
     if (A > 0){
        soma = soma + A;
        cont = cont + 1:
     }
  }while (A != 0);
  printf("Soma = %d  e  quantidade = %d\n", soma, cont);
}
```

4.

Implementar um **subprograma** em C que dado dois números inteiros positivos não nulo **N1** e **N2** (N1, N2 > 0), com **N1 < N2** (N1 e N2 parâmetros do subprograma), **determine e devolva** como resultados a **soma dos números pares** e a **soma dos números ímpares** existentes entre **N1** e **N2**, inclusivé (em {N1, …, N2}).

```
// Versão 1: devolve as somas de impares e de pares pelo cabeçalho
void somaParesImpares (int N1, int N2, int *pares, int *impares){
  int  k;
  *pares = 0;
  *impares = 0;
  for (k = N1; k <= N2; k++){
     if (k % 2 == 0)
        *pares = *pares + k;
     else
        *impares = *impares + k;
  }
}

// Versão 2: devolve a somas de impares pelo return e a soma de pares pelo cabeçalho
int somaParesImpares (int N1, int N2, int *pares){
  int  k, impares;
  *pares = 0;
  impares = 0;
  for (k = N1; k <= N2; k++){
     if (k % 2 == 0)
        *pares = *pares + k;
     else
        impares = impares + k;
  }
  return impares;
}
```

5. [1.0 val]

Considere as seguintes declarações de variáveis:

int *V, **W, *X;

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

(esquema de um bloco de memória)

	...	
100500	110172	**X**
	...	
100700	110168	**V**
	...	
110160	100	
110164	900	
110168	500	
110172	1000	
110176	1500	
110180	50	
	...	
110500	100700	**w**
	...	

Usando os valores contidos no esquema de um bloco de memória dado ao lado, indique, justificando, os valores de cada uma das seguintes expressões:

a) *(X + 2)

b) V - 4

c) W[0]

d) &(*(X - 1))

e) *X - 1

f) W + 2

g) *(*W - 2)

h) *V + 3

NOTA: Caso o valor da expressão não exista ou seja desconhecido, deve responder "**Indefinido**".

a) *(X + 2) = *(110172 + 2 x 4) = *(110180) = **50**

b) V – 4 = 110168 – 4 x 4 = **110152**

c) W[0] = **110168**

d) &(*(X – 1)) = &(*(110172 – 1 x 4)) &(*(110168)) = **110168**

e) *X – 1 = 1000 – 1 = **999**

f) W + 2 = 100700 + 2 x 8 = **100716**

g) *(*W – 2) = *(110168 – 2 x 4) = *(110160) = **100**

h) *V + 3 = 500 + 3 = **503**

6. [2.5 val]

Implementar um **subprograma** em C que dado um array de 1 dimensão **X** com **N** números inteiros (X e N são parâmetros do subprograma), **determine e devolva** o **segundo menor número** que existe em **X**.

Exemplo: X = [21 7 2 21 -14 2 -14 23], segundo menor número é o **2** (o menor número é -14)

```
int segundoMenorNumero (int X[], int N){
  int k, menor, segMenor;
  menor = X[0];
  segMenor = X[0];
  for (k = 1; k < N; k++)
     if (X[k] < menor){
        segMenor = menor;
        menor = X[k];
     }
     else
        if (X[k] < segMenor)
           segMenor = X[k];
  return segMenor;
}
```

7. [4.0 val]

Considere a seguinte definição de um tipo estrutura associada aos dados de uma pessoa:

```
typedef struct {
      int     numCC;      // Número de Cartão de Cidadão
      int     anoNasc;    // Ano de nascimento (ex: 1986)
      float   altura;     // Altura em metros (exemplo: 1.85)
} PESSOA;
```

Considere também o seguinte subprograma que se encontra já implementado na biblioteca "Exame.h":

```
void maiorAltura (PESSOA X[], float *maior);
```

que **devolve** como resultado o **maior valor do campo altura** dos elementos (registos) do array **X**.

Seja o ficheiro de texto **"Pessoas.txt"** com dados até no máximo 500 pessoas; cada linha do ficheiro guarda os dados de uma única pessoa (por esta ordem): número de CC, ano de nascimento e altura.

Implemente um **programa em C** que realize as seguintes acções (pela ordem indicada):

- com os dados do ficheiro **"Pessoas.txt"**, **construa** um **array de 1 dimensão X** do tipo **PESSOA** com **todos** os dados do ficheiro,
- determine o **maior valor** da **altura** dos elementos do array **X**, **maior** (usar subprograma dado),
- peça ao utilizador um número real **ALT** menor ou igual a **maior** (**0 < ALT ≤ maior**),
- **guarde** no ficheiro **"Altura.txt"** todos os elementos do array **X** cuja valor do campo **altura** é maior ou igual a **ALT** (≥ ALT)

```
#include <stdio.h>
typedef struct {
      int  numCC;
      int  anoNasc;
      float  altura;
} PESSOA;
#include "Exame.h"  // o subprograma maiorAltura precisa do tipo de dados PESSOA

main(){
  PESSOA  P[500];
  int  TAM, k, num, ano;
  float  alt, maior, ALT;
  FILE  *f, *g;

  TAM = 0;
  f = fopen("Pessoas.txt", "r");
  while (fscanf(f, "%d%d%f", &num, &ano, &alt) == 3){
     TAM = TAM + 1;
     P[TAM-1].numCC = num;
     P[TAM-1].anoNasc = ano;
     P[TAM-1].altura = alt;
  }
  fclose(f);

  maiorAltura(P, &maior);

  do{
     printf("Insira um real > 0 e <= %d: ", maior);
     scanf("%f", &ALT);
  }while(ALT <= 0  ||  ALT > maior);

  g = fopen("Altura.txt", "w");
  for(k = 0; k < TAM; k++)
     if (P[k].altura >= ALT)
        fprintf(g, "%d %d %f\n", P[k].numCC, P[k].anoNac, P[k].altura);
  fclose(g);
}
```