UNIVERSIDADE DA BEIRA INTERIOR

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

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

1.

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

(a) atribua à variável **B** a parte fracionária do valor da variável do tipo real **X** (Ex: X = 2.65; B = 0.65)

(b) atribua à variável **P** o valor **10** se o valor da variável do tipo inteiro **N** for par e **20** se **N** for ímpar

(c) atribuir à variável **CENT** o algarismo das centenas da variável do tipo inteiro **A**

(a) B = X – (int) X;

(b) P = 10 + (N % 2) * 10;

(c) CENT = (A / 100) % 10;

Supondo que **X = 10**, **Y = -15** 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	**+** 2	**&&** X	**!=** -2	***** Z
(e)	Y **+** X	***** (0.5	***** 6	**/** 4)	**/** 3	
(f)	Z **+** (4	**+** Y)	***** (4	**/** Z)	**+** X	

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

resultado: 1 (Verdadeiro/True)

```
-15 < 5 * 3 + 2 && 10 != -2 * 5
-15 < 15 + 2 && 10 != -2 * 5        (após aplicar o operador 2)
-15 < 15 + 2 && 10 != -10           (após aplicar o operador 6)
-15 < 17 && 10 != -10               (após aplicar o operador 3)
1(V) && 10 != -10                   (após aplicar o operador 1)
1(V) && 1(V)                        (após aplicar o operador 5)
1(V)                                (após aplicar o operador 4)
```

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

resultado: -12.5

```
-15 + 10 * ( 0.5 * 6 / 4 ) / 3
-15 + 10 * ( 3.0 / 4 ) / 3          (após aplicar o operador 3)
-15 + 10 * 0.75 / 3                 (após aplicar o operador 4)
-15 + 7.5 / 3                       (após aplicar o operador 2)
-15 + 2.5                           (após aplicar o operador 5)
-12.5                               (após aplicar o operador 1)
```

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

resultado: 15

```
5 + ( 4 + -15 ) * ( 4 / 5 ) + 10
5 + -11 * ( 4 / 5 ) + 10            (após aplicar o operador 2)
5 + -11 * 0 + 10                    (após aplicar o operador 4)
5 + 0 + 10                          (após aplicar o operador 3)
5 + 10                              (após aplicar o operador 1)
15                                  (após aplicar o operador 5)
```

2.

Construa um algoritmo, usando um fluxograma, que:

- peça ao utilizador e insira/leia um número inteiro N positivo não nulo (N > 0);
- peça ao utilizador e insira/leia N números inteiros;
- determine o **maior** número inserido (dos N números inseridos):
- mostre o resultado obtido (maior número inserido).

3.

Construa um **programa em C** que:

- peça ao utilizador e insira/leia um número inteiro N positivo não nulo (N > 0);
- peça ao utilizador e insira/leia N números inteiros;
- determine o **maior** número inserido (dos N números inseridos):
- mostre o resultado obtido (maior número inserido).

```
#include <stdio.h>
main(){
    int  N, A, maior, k;
    do{
       printf("Insira um inteiro > 0: ");
       scanf("%d", &N);
    }while (N <= 0);
    printf("Insira o primeiro numero inteiro: ");
    scanf("%d", &A);
    maior = A;
    for (k = 2; k <= N; k++){
       printf("Insira outro numero inteiro: ");
       scanf("%d", &A);
       if (A > maior)
          maior = A;
    }
    printf("Maior = %d\n", maior);
}
```

4.

Implementar um **subprograma** em C que dado um número inteiro positivo não nulo **N** (parâmetro do subprograma), **determine e devolva** a **soma** dos algarismos que formam o número **N**.

```
int somaAlgarismos (int N){
    int  soma;
    soma = 0;
    while (N != 0){
       soma = soma + N % 10;
       N = N / 10;
    }
    return soma;
}
```

5.

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	110168	**X**
	...	
100700	110164	**V**
	...	
110160	100	
110164	900	
110168	500	
110172	1000	
110176	1500	
110180	50	
	...	
110500	100500	**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) *(V + 3)

b) V - 3

c) &V[4]

d) &(X - 1)

e) **W - 2

f) W + 4

g) *(*W + 3)

h) *W - 2

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

a) *(V + 3) = *(110164 + 3 x 4) = *(110176) = **1500**

b) V – 3 = 110164 – 3 x 4 = **110152**

c) &V[4] = **110180**

d) &(X – 1) = 110164 ---> &(110168 – 1x4) = &(110164) => **INDEFINIDO**

e) **W – 2 = 500 – 2 = **498** (W = 100500 => *W = 110168 => **W = 500)

f) W + 4 = 100500 + 4 x 8 = **100532**

g) *(*W + 3) = *(110168 + 3 x 4) = *(110180) = **50**

h) *W – 2 = 110168 – 2 x 4 = **110160**

6.

Implementar um **subprograma** em C que dados um array (de 1 dimensão) **X** com **N** números reais (X e N são parâmetros do subprograma), **determine e devolva** como resultados a quantidade (número inteiro) e a média aritmética (número real) dos números **positivos de X**. Caso o array X não contenha números positivos, os resultados a devolver devem ser valores nulos (valor 0 para ambos os resultados).

Versão 1:

```
void quantidadeMediaPositivos  (float X[], int N, int *quant, float *media){
    int k;
    float soma;
    *quant = 0;
    soma = 0;
    for (k = 0; k < N; k++)
        if (X[k] > 0){
            *quant = *quant + 1;
            soma = soma + X[k];
        }
    if (*quant > 0)
        *media = soma / *quant;
    else
        *media = 0;
}
```

Versão 2:

```
int quantidadeMediaPositivos  (float X[], int N, float *media){
    int k, quant;
    float soma;
    quant = 0;
    soma = 0;
    for (k = 0; k < N; k++)
        if (X[k] > 0){
            quant = quant + 1;
            soma = soma + X[k];
        }
    if (quant > 0)
        *media = soma / quant;
    else
        *media = 0;
    return quant;
}
```

7.

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 decadaAno (int A, int *D);
```

que determina e devolve como resultado a década a que pertence o ano A.

Seja o ficheiro de texto **"Pessoas.txt"** com informação de 200 pessoas, em que 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):

- peça ao utilizador um número inteiro **A** entre 1900 e 2023, correspondente a um ano (ex: 1987),
- determine a década **D** a que pertence o ano **A** (usar subprograma dado),
- com os dados do ficheiro **"Pessoas.txt"**, **construa** um **array de 1 dimensão** do tipo **PESSOA P apenas** com os dados das pessoas nascidas na década **D** (ex: D = 1980; 1980 ≤ **anoNasc** ≤ 1989),
- **guarde** no ficheiro **"Decada.txt"** todos os elementos do array **P** construído antes.

```
#include <stdio.h>
#include "Exame.h"
typedef struct {
    int  numCC;
    int  anoNasc;
    float  altura;
} PESSOA;
main(){
  PESSOA  P[200];
  int  TAM, k, A, D, num, ano, k;
  float alt;
  FILE  *f, *g;
  do{
     printf("Insira um inteiro entre 1900 e 2023: ");
     scanf("%d", &A);
  }while(A < 1900  ||  A > 2023);
  decadaAno(A, &D);     //  ATENÇÃO
  TAM = 0;
  f = fopen("Pessoas.txt", "r");
  while (fscanf(f, "%d%d%f", &num, &ano, &alt) == 3)
     if (ano >= D  &&  ano <= D + 9){
        TAM = TAM + 1;
        P[TAM-1].numCC = num;    P[TAM-1].anoNasc = ano;    P[TAM-1].altura = alt;
     }
  }
  fclose(f);
  g = fopen("Decada.txt", "w");
  for(k = 0; k < TAM; k++)
     fprintf(g, "%d %d %f\n", P[k].numCC, P[k].anoNac, P[k].altura);
  fclose(g);
}
```