

UNIVERSIDADE DA BEIRA INTERIOR

Programação – LEI

1º Semestre

Exame Época Recurso

Resolução

2024/2025

1.

Escreva uma **expressão lógica** em linguagem C para as seguintes condições:

a) o valor da variável do tipo real **A não pertence** ao intervalo **[-50, 50[**

$A < -50 \ || \ A \geq 50$

b) o valor da variável do tipo inteiro **B é positivo e não pertence** ao conjunto **{ 100, ..., 999 }**

$B \geq 0 \ \&\& \ (B < 100 \ || \ B > 999)$

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

c) a variável **B** recebe o **valor** do número composto pelos **três últimos algarismos** do valor (número) da variável do tipo inteiro **A** (exemplo: $A = 23542 \Rightarrow B = 542$)

$B = A \% 1000;$

d) a variável **P** recebe o valor **5** se o valor da variável **N** for **par** e o valor **9** se **N** for **ímpar** (P e N são variáveis do tipo inteiro)

$P = (N \% 2) * 4 + 5;$

Supondo que **X = 12**, **Y = 2** e **Z = 6** (X, Y e Z são variáveis do tipo inteiro), indique a ordem de cálculo dos operadores e determine o valor de cada uma das seguintes expressões. Apresente todos os cálculos.

e) $(Y \underset{1}{+} 5 \underset{2}{*} Z) \underset{3}{<} X \underset{4}{/} (Z \underset{5}{\%} 4)$

ordem de cálculo: **2, 1, 5, 4, 3**

valor: **falso**

$(2 + 5 * 6) < 12 / (6 \% 4) =$ (2)

$(2 + 30) < 12 / (6 \% 4) =$ (1)

$32 < 12 / (6 \% 4) =$ (5)

$32 < 12 / 2 =$ (4)

$32 < 6 =$ (3)

falso

f) $(3 \underset{1}{+} 2.5) \underset{2}{*} Y \underset{3}{-} (24 \underset{4}{\%} (X \underset{5}{+} Z))$

ordem de cálculo: **1, 5, 4, 2, 3**

valor: **5.0**

$(3 + 2.5) * 2 - (24 \% (12 + 6)) =$ (1)

$5.5 * 2 - (24 \% (12 + 6)) =$ (5)

$5.5 * 2 - (24 \% 18) =$ (4)

$5.5 * 2 - 6 =$ (2)

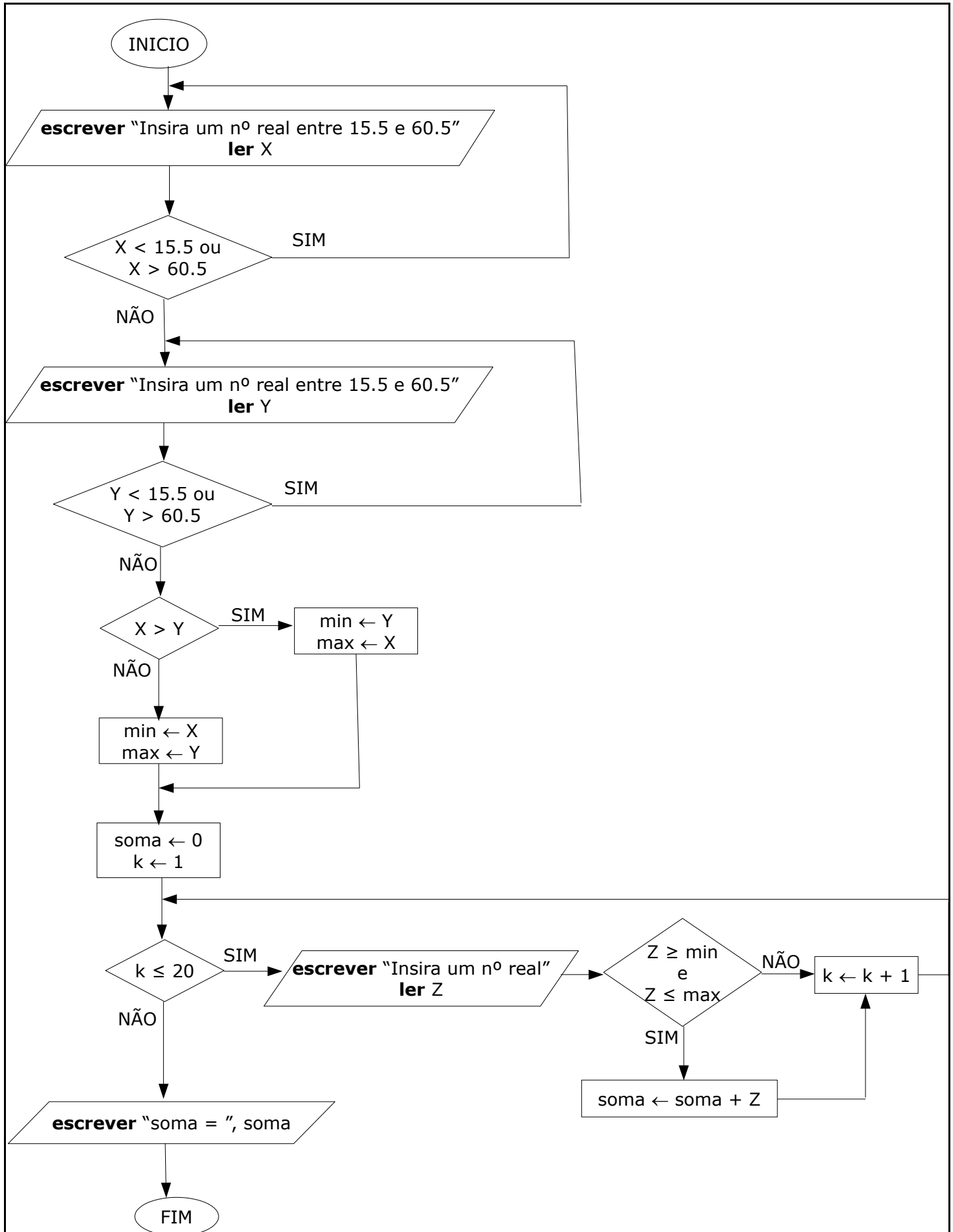
$11.0 - 6 =$ (3)

5.0

2.

Construa um algoritmo (**sem usar arrays**), através de um fluxograma, que:

- **peça** ao utilizador e **insira/leia** dois números **reais**, **X** e **Y**, com valores no intervalo **[15.5, 60.5]**
- **determine** os valores de **min** = menor(X, Y) e **max** = maior(X, Y)
- **peça** ao utilizador e **insira/leia** **20** números **reais** e **determine** a **soma** dos números inseridos que **pertencem** ao intervalo **[min, max]**
- **mostre** o **resultado obtido** no passo anterior (**soma**).



3.

Construa um **programa em C (sem usar arrays)** que:

- **peça** ao utilizador e **insira/leia** dois números **reais, X e Y**, com valores no intervalo **[15.5, 60.5]**
- **determine** os valores de **min** = menor(X, Y) e **max** = maior(X, Y)
- **peça** ao utilizador e **insira/leia 20** números **reais** e **determine** a **soma** dos números inseridos que **pertencem** ao intervalo **[min, max]**
- **mostre** o **resultado obtido** no passo anterior (**soma**).

```
#include <stdio.h>

int main()
{
    float X, Y, min, max, Z, soma;
    int k;
    do {
        printf("Inserir um numero real entre 15.5 e 60.5: ");
        scanf("%f", &X);
    } while (X < 15.5 || X > 60.5);
    do {
        printf("Inserir um numero real entre 15.5 e 60.5: ");
        scanf("%f", &Y);
    } while (Y < 15.5 || Y > 60.5);
    if (X > Y) {
        min = Y;
        max = X;
    }
    else {
        min = X;
        max = Y;
    }
    soma = 0;
    for (k = 1; k <= 20; k++) {
        printf("Inserir um numero real: ");
        scanf("%f", &Z);
        if (Z >= min && Z <= max)
            soma = soma + Z;
    }
    printf("Soma = %f\n", soma);
}
```

4.

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 o produto** entre todos os números **positivos não nulos** (> 0) **de X** e a **soma** dos números **negativos de X**.

NOTA: não pode usar outros arrays nem estruturas.

```
// devolve pelo return a soma dos negativos e pelo cabeçalho o produto dos positivos não nulos
float produtoSoma (float X[], int N, float *produto)
{
    int k;
    float soma;
    soma = 0;
    *produto = 1;
    for (k = 0; k < N; k++) {
        if (X[k] > 0)
            *produto = *produto * X[k];
        else
            if (X[k] < 0)
                soma = soma + X[k];
    }
    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)

	...	
110040	110252	X
	...	
110120	110256	V
	...	
110244	90	
110248	80	
110252	70	
110256	60	
110260	50	
110264	40	
	...	
110500	110120	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)** $*(V - 2)$ = $*(110256 - 2 \times \text{sizeof(int)}) = *(110256 - 8) = *(110248) = \mathbf{80}$
- b)** $V + 5$ = $110256 + 5 \times \text{sizeof(int)} = 110256 + 20 = \mathbf{110276}$
- c)** $\&V[1]$ = **110260**
- d)** $*(X - 2)$ = $*(110252 - 2 \times \text{sizeof(int)}) = *(110252 - 8) = *(110244) = \mathbf{90}$
- e)** $**W + 4$ = $60 + 4 = \mathbf{64}$
- f)** $W + 4$ = $110120 + 4 \times \text{sizeof(int*)} = 110120 + 32 = \mathbf{110152}$
- g)** $*(W + 2)$ = $*(110256 + 2 \times \text{sizeof(int)}) = *(110256 + 8) = *(110264) = \mathbf{40}$
- h)** $*W - 2$ = $110256 - 2 \times \text{sizeof(int)} = 110256 - 8 = \mathbf{110248}$
- i)** $W[0]$ = **110256**
- j)** $X[2]$ = **50**

NOTA: se não existir resposta, indicar com **ERRO**

6.

Considere um ficheiro de texto de nome "**dados.txt**", em que **cada linha** deste ficheiro contém **dois** números **inteiros**. Implemente um **programa em C** que realize as seguintes acções, pela ordem indicada:

1º) construa um **array 1D** com os **números** contidos no ficheiro "**dados.txt**", usando **gestão dinâmica de memória**, da seguinte forma: **para cada linha** do ficheiro, **leia os dois números** e **acrescente ao array os dois números** (se os dois números são diferentes entre si) ou **acrescente apenas um** deles (se os dois números são iguais),

2º) guarde os elementos do **array** que são **positivos** no ficheiro de texto "**saida.txt**".

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

int main()
{
    int *A, num1, num2, tamA, k;
    FILE *f;
    // 1º)
    f = fopen("dados.txt", "r");
    tamA = 0;
    X = (int*) malloc(tamX * sizeof(int));
    while (fscanf(f, "%d%d", &num1, &num2) == 2) {
        if (num1 != num2) {
            tamA = tamA + 2;
            A = (int*) realloc(A, tamA * sizeof(int));
            A[tamA-2] = num1;
            A[tamA-1] = num2;
        }
        else {
            tamA = tamA + 1;
            A = (int*) realloc(A, tamA * sizeof(int));
            A[tamA-1] = num1;
        }
    }
    fclose(f);
    // 2º)
    f = fopen("saida.txt", "w");
    for (k = 0; k < tamA; k++) {
        if (A[k] >= 0)
            fprintf(f, "%d\n", A[k]);
    }
    fclose(f);
}
```

7.

Implementar uma **função recursiva** em C que dados um **array 1D X** com **N** números **reais** (com X e N parâmetros da função), **determine** a **soma** dos números **positivos** do **array X**.

```
float somaPositivos (float *X, int N)
{
    float soma;
    // caso base/terminal
    if (N == 0)
        return 0;
    // caso geral
    soma = somaPositivos(X, N-1);
    if (X[N-1] >= 0)
        return soma + X[N-1];
    else
        return soma;
}
```

8.

Implementar um **subprograma** em C que dados um **array 1D A** com **N** números **inteiros** (A e N > 0 são parâmetros do subprograma), **remova** do array **A** todos os **números que se repetem**, ficando o array **sem repetidos** (mantendo ou não a ordem inicial entre os números).

NOTA: não pode usar outros arrays nem ficheiros, e deve percorrer o array o número mínimo de vezes.

Exemplo: A = [2 -4 9 -4 2 5 9 -4 7] => A = [2 -4 9 2 5 9 -4 7]

```
int* removerRepetidos (int *A, int *N)
{
    int k, j;
    k = 0;
    while (k < *N) {
        j = k + 1;
        while (j < *N) {
            if (A[j] == A[k]) {
                A[j] = A[*N-1];
                *N = *N - 1;
            }
            else
                j = j + 1;
        }
        k = k + 1;
    }
    A = (int *) realloc (A, *N * sizeof(int));
    return A;
}
```