UNIVERSIDADE DA BEIRA INTERIOR

Programação – LEI **1º Semestre**

Exame Época de Recurso (16 val) 2h + 15 min **2023/2024**

1. [1.50 val]

Escreva uma **expressão lógica em linguagem C** para a seguinte condição:

a) metade do valor da variável do tipo inteiro **N** é pelo menos igual ao resto da divisão de N por 5

b) o valor da variável do tipo real **X** não pertence ao intervalo [-5, 10]

c) o valor da variável do tipo inteiro **N** é no mínimo 50 e no máximo 100, e não é múltiplo de 20

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

d) o valor da variável do tipo inteiro **A** é aumentado para o seu triplo

e) a variável do tipo real **Z** recebe o valor da soma das partes inteiras das variáveis do tipo real **X** e **Y**

f) a variável do tipo inteiro **P** recebe o valor **1** se **N** for par ou **-1** se **N** for ímpar

g) a variável do tipo inteiro **SOMA** recebe o valor da soma do algarismo das **dezenas** do valor da variável **A** com o algarismo dos **milhares** do valor da variável **B** (A e B são do tipo inteiro)

2. [2.50 val]

Usando um fluxograma, construa um algoritmo que realize as seguintes ações (pela ordem indicada):

1. peça ao utilizador que insira um número inteiro N entre 10 e 50, inclusivé
2. peça ao utilizador que insira N números reais e determine a **quantidade** de números positivos (> 0) inseridos e a **soma** dos números positivos (> 0) inseridos
3. mostre os resultados obtidos antes

3. [2.50 val]

Construa um **programa em C** que realize as seguintes ações (pela ordem indicada):

1. peça ao utilizador que insira um número inteiro N entre 1 e 50, inclusivé
2. peça ao utilizador para inserir N números inteiros não nulos (≠ 0) e, determine a **quantidade** de números positivos (> 0) **pares** e a **quantidade** de números positivos (> 0) **ímpares**
3. mostre os resultados obtidos antes (quantidade de pares e quantidade de ímpares)

4. [2.25 val]

Implemente um **subprograma em C** que dado um array de 1 dimensão **X** com **N** (N > 0) números reais positivos (> 0) e um valor real positivo **K** (X, N e K são parâmetros do subprograma), **determine** e **devolva** como resultados a **quantidade** de elementos de X que são menores do que K e o **maior** elemento de **X** que é menor do que K.

NOTA: Se todos os elementos de X forem maiores do que K, então considerar maior = -1

5. [1.50 val]

Considere as seguintes declarações de variáveis:

int *V, **W, *X;

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

(esquema de bloco de memória)

	...	
100350	110144	**V**
	...	
100725	100350	**W**
	...	
110132	12	
110136	15	
110140	16	
110144	13	
110148	14	
110152	17	
	...	
110585	110136	**X**
	...	

Considerando os valores que constam no esquema de um bloco de memória que se encontra ao lado, determine o valor de cada uma das seguintes expressões (**apresentar todos os cálculos efetuados**):

a) X + 2

b) V[0]

c) W + 4

d) (*W)[1]

e) V + 2

f) *V + 3

g) **W + 4

h) W[0] + 4

i) X – 2

j) *(*W – 2)

NOTA: se o valor da expressão não existir ou for desconhecido, então responder "**Indefinido**"

6. [2.50 val]

Considere o ficheiro de texto "reais.txt" apenas com números reais. Implemente um **programa em C** que realize as seguintes ações (pela ordem indicada):

1. usando gestão de **memória dinâmica**, **construa** dois **arrays de 1 dimensão**, um com todos os números positivos (> 0) e um outro com todos os números negativos (< 0) do ficheiro "reais.txt"
2. **guarde** no ficheiro de texto "saida.txt" todos os elementos do array de negativos construído em 1
3. **acrescente** ao ficheiro de texto "saida.txt" todos os elementos do array de positivos construído em 1

NOTA: não é preciso verificar se há **ERRO** na **abertura dos ficheiros**, nem na **alocação de memória**

7. [3.25 val]

Implemente um **subprograma em C** que dado um **array** de 1 dimensão **X** com **N** (N > 0) números inteiros (X e N são parâmetros do subprograma), **remova** do array todos os elementos que **não são repetidos** (são únicos ou só aparecem 1 vez) e mantendo a ordem pela qual se encontram no array, tendo em conta que o array foi construído usando memória dinâmica.

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

Exemplo:

A = [3 -5 2 8 3 4 8 6 -2 -5] ==> A = [3 -5 8 3 8 -5]