Programação Imperativa

1º Ano - LEI/LCC

Questões 1ª Parte

- 1. Defina um programa que lê (usando a função scanf uma sequência de números inteiros terminada com o número 0 e imprime no ecran a soma dos números lidos.
- 2. Defina um programa que lê (usando a função scanf uma sequência de números inteiros terminada com o número 0 e imprime no ecran o maior elemento da sequência.
- 3. Defina um programa que lê (usando a função scanf uma sequência de números inteiros terminada com o número 0 e imprime no ecran a média da sequência.
- 4. Defina um programa que lê (usando a função scanf uma sequência de números inteiros terminada com o número 0 e imprime no ecran o segundo maior elemento.
- 5. Defina uma função int bitsUm (unsigned int n) que calcula o número de bits iguais a 1 usados na representação binária de um dado número n.
- 6. Defina uma função int trailing (unsigned int n) que calcula o número de bits a 0 no final da representação binária de um número (i.e., o expoente da maior potência de 2 que é divisor desse número).
- 7. Defina uma função int qDig (unsigned int n) que calcula o número de dígitos necessários para escrever o inteiro n em base decimal. Por exemplo, qDig (440) deve retornar 3.
- 8. Apresente uma definição da função pré-definida em C int strlen (char str[]) que calcula o comprimento de uma string.
- 9. Apresente uma definição da função pré-definida em C char *strcat (char s1[], char s2[]) que concatena a string s2 a s1 (retornando o endereço da primeira).
- 10. Apresente uma definição da função pré-definida em C char *strcpy (char *dest, char source[]) que copia a string source para dest retornando o valor desta última.
- 11. Apresente uma definição da função pré-definida em C int strcmp (char s1[], char s2[]) que compara (lexicograficamente) duas strings. O resultado deverá ser
 - 0 se as strings forem iguais
 - <0 se s1 < s2
 - \bullet >0 se s1 > s2

- 12. Apresente uma definição da função pré-definida em C char *strstr (char s1[], char s2[]) que determina a posição onde a string s2 ocorre em s1. A função deverá retornar NULL caso s2 não ocorra em s1.
- 13. Defina uma função void strrev (char s[]) que inverte uma string.
- 14. Defina uma função void strnoV (char s[]) que retira todas as vogais de uma string.
- 15. Defina uma função void truncW (char t[], int n) que dado um texto t com várias palavras (as palavras estão separadas em t por um ou mais espaços) e um inteiro n, trunca todas as palavras de forma a terem no máximo n caracteres. Por exemplo, se a string txt contiver "liberdade, igualdade e fraternidade", a invocação de truncW (txt, 4) deve fazer com que passe a estar lá armazenada a string "libe igua e frat".
- 16. Defina uma função char charMaisfreq (char s[]) que determina qual o caracter mais frequente numa string. A função deverá retornar 0 no caso de s ser a string vazia.
- 17. Defina uma função int iguaisConsecutivos (char s[]) que, dada uma string s calcula o comprimento da maior sub-string com caracteres iguais. Por exemplo, iguaisConsecutivos ("aabcccaac") deve dar como resultado 3, correspondendo à repetição "ccc".
- 18. Defina uma função int difConsecutivos (char s[]) que, dada uma string s calcula o comprimento da maior sub-string com caracteres diferentes. Por exemplo, difConsecutivos ("aabcccaac") deve dar como resultado 3, correspondendo à string "abc".
- 19. Defina uma função int maiorPrefixo (char s1 [], char s2 []) que calcula o comprimento do maior prefixo comum entre as duas strings.
- 20. Defina uma função int maiorSufixo (char s1 [], char s2 []) que calcula o comprimento do maior sufixo comum entre as duas strings.
- 21. Defina a função int sufPref (char s1[], char s2[]) que calcula o tamanho do maior sufixo de s1 que é um prefixo de s2. Por exemplo sufPref("batota", "totalidade") deve dar como resultado 4, uma vez que a string "tota" é um sufixo de "batota" e um prefixo de "totalidade".
- 22. Defina uma função int contaPal (char s[]) que conta as palavras de uma string. Uma palavra é uma sequência de caracteres (diferentes de espaço) terminada por um ou mais espaços. Assim se a string p tiver o valor "a a bb a", o resultado de contaPal (p) deve ser 4.
- 23. Defina uma função int contaVogais (char s[]) que retorna o número de vogais da string s. Não se esqueça de considerar tanto maiúsculas como minúsculas.
- 24. Defina uma função int contida (char a[], char b[]) que testa se todos os caracteres da primeira string também aparecem na segunda. Por exemplo, contida "braga" "bracara augusta" deve retornar *verdadeiro* enquanto que contida "braga" "bracarense" deve retornar *falso*.
- 25. Defina uma função int palindorome (char s[]) que testa se uma palavra é palíndrome, i.e., lê-se de igual forma nos dois sentidos.

- 26. Defina uma função int remRep (char x[]) que elimina de uma string todos os caracteres que se repetem sucessivamente deixando lá apenas uma cópia. A função deverá retornar o comprimento da string resultante. Assim, por exemplo, ao invocarmos a função com uma vector contendo "aaabaaabbbaaa", o vector deve passar a conter a string "ababa" e a função deverá retornar o valor 5.
- 27. Defina uma função int limpaEspacos (char t[]) que elimina repetições sucessivas de espaços por um único espaço. A função deve retornar o comprimento da string resultante.
- 28. Defina uma função void insere (int v[], int N, int x) que insere um elemento (x) num vector ordenado. Assuma que as N primeiras posições do vector estão ordenadas e que por isso, após a inserção o vector terá as primeiras N+1 posições ordenadas.
- 29. Defina uma função void merge (int r [], int a[], int b[], int na, int nb) que, dados vectores ordenados a (com na elementos) e b (com nb elementos), preenche o vector r (com na+nb elementos) com os elementos de a e b ordenados.
- 30. Defina uma função int crescente (int a[], int i, int j) que testa se os elementos do vector a, entre as posições i e j (inclusivé) estão ordenados por ordem crescente. A função deve retornar 1 ou 0 consoante o vector esteja ou não ordenado.
- 31. Defina uma função int retiraNeg (int v[], int N) que retira os números negativos de um vector com N inteiros. A função deve retornar o número de elementos que não foram retirados.
- 32. Defina uma função int menosFreq (int v[], int N) que recebe um vector v com N elementos ordenado por ordem crescente e retorna o menos frequente dos elementos do vector. Se houver mais do que um elemento nessas condições deve retornar o que começa por aparecer no índice mais baixo.
- 33. Defina uma função int maisFreq (int v[], int N) que recebe um vector v com N elementos ordenado por ordem crescente e retorna o mais frequente dos elementos do vector. Se houver mais do que um elemento nessas condições deve retornar o que começa por aparecer no índice mais baixo.
- 34. Defina uma função int maxCresc (int v[], int N) que calcula o comprimento da maior sequência crescente de elementos consecutivos num vector v com N elementos. Por exemplo, se o vector contiver 10 elementos pela seguinte ordem: 1, 2, 3, 2, 1, 4, 10, 12, 5, 4, a função deverá retornar 4, correspondendo ao tamanho da sequência 1, 4, 10, 12.
- 35. Defina uma função int elimRep (int v[], int n) que recebe um vector v com n inteiros e elimina as repetições. A função deverá retornar o número de elementos do vector resultante. Por exemplo, se o vector v contiver nas suas primeiras 10 posições os números

$$\{1, 2, 3, 2, 1, 4, 2, 4, 5, 4\}$$

a invocação elimRep (v,10) deverá retornar 5 e colocar nas primeiras 5 posições do vector os elementos {1,2,3,4,5}.

- 36. Defina uma função int elimRepOrd (int v[], int n) que recebe um vector v com n inteiros ordenado por ordem crescente e elimina as repetições. A função deverá retornar o número de elementos do vector resultante.
- 37. Defina uma função int comunsOrd (int a[], int na, int b[], int nb) que calcula quantos elementos os vectores a (com na elementos) e b (com nb elementos) têm em comum. Assuma que os vectores a e b estão ordenados por ordem crescente.
- 38. Defina uma função int comuns (int a[], int na, int b[], int nb) que calcula quantos elementos os vectores a (com na elementos) e b (com nb elementos) têm em comum. Assuma que os vectores a e b não estão ordenados e defina a função sem alterar os vectores.
- 39. Defina uma função int minInd (int v[], int n) que, dado um vector v com n inteiros, retorna o índice do menor elemento do vector.
- 40. Defina uma função void somasAc (int v[], int Ac [], int N) que preenche o vector Ac com as somas acumuladas do vector v. Por exemplo, na posição Ac[3] deve ser calculado como v[0]+v[1]+v[2]+v[3].
- 41. Defina uma função int triSup (int N, float m [N][N]) que testa se uma matriz quadrada é triangular superior, i.e., que todos os elementos abaixo da diagonal são zeros.
- 42. Defina uma função void transposta (int N, float m [N][N]) que transforma uma matriz na sua transposta.
- 43. Defina uma função void addTo (int N, int M, int a [N][M], int b[N][M]) que adiciona a segunda matriz à primeira.
- 44. Escreva em C uma função void sumDiag (int N, int m [N][N]) que recebe como argumento uma matriz quadrada e substitui cada elemento da diagonal pelo somatório de todos os outros elementos dessa linha. O exemplo abaixo mostra o efeito desta função numa matriz 4×4 .

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \implies \begin{bmatrix} 9 & 2 & 3 & 4 \\ 5 & 20 & 7 & 8 \\ 9 & 10 & 31 & 12 \\ 13 & 14 & 15 & 42 \end{bmatrix}$$

- 45. Escreva um programa que liste no ecran as letras do alfabeto (maiúsculas e minúsculas) e o respectivo código ASCII. Use para isso a função printf, tanto para imprimir os carateres como os seus códigos (inteiros).
- 46. Uma forma de representar conjuntos de índices consiste em usar um array de inteiros contendo 1 ou 0 consoante esse índice pertença ou não ao conjunto. Assim o conjunto {1,4,7} seria representado por um array em que as primeiras 8 posições conteriam {0,1,0,0,1,0,0,1}. Apresente uma definição da função int unionSet (int N, int v1[N], int v2[N], int r[N]) que coloca no array r o resultado da união dos conjuntos v1 e v2.
- 47. Uma forma de representar conjuntos de índices consiste em usar um array de inteiros contendo 1 ou 0 consoante esse índice pertença ou não ao conjunto. Assim o conjunto {1,4,7} seria representado por um array em que as primeiras 8 posições conteriam {0,1,0,0,1,0,0,1}.

- Apresente uma definição da função int intersectSet (int N, int v1[N], int v2[N], int r[N]) que coloca no array r o resultado da intersecção dos conjuntos v1 e v2.
- 48. Uma forma de representar multi-conjuntos de índices consiste em usar um array de inteiros contendo em cada posição o número de ocorrências desse índice. Assim o multi-conjunto {1,1,4,7,7,7} seria representado por um array em que as primeiras 8 posições conteriam {0,2,0,0,1,0,0,3}.
 - Apresente uma definição da função int intersectMSet (int N, int v1[N], int v2[N], int r[N]) que coloca no array r o resultado da intersecção dos multi-conjuntos v1 e v2.
- 49. Uma forma de representar multi-conjuntos de índices consiste em usar um array de inteiros contendo em cada posição o número de ocorrências desse índice. Assim o multi-conjunto {1,1,4,7,7,7} seria representado por um array em que as primeiras 8 posições conteriam {0,2,0,0,1,0,0,3}.
 - Apresente uma definição da função int unionMSet (int N, int v1[N], int v2[N], int r[N]) que coloca no array r o resultado da união dos multi-conjuntos v1 e v2.
- 50. Uma forma de representar multi-conjuntos de índices consiste em usar um array de inteiros contendo em cada posição o número de ocorrências desse índice. Assim o multi-conjunto {1,1,4,7,7,7} seria representado por um array em que as primeiras 8 posições conteriam {0,2,0,0,1,0,0,3}.
 - Apresente uma definição da função int cardinalMSet (int N, int v[N]) que calcula a número de elementos do multi-conjunto v.