Exercícios de Linguagem C

Aula 1 – Aspectos básicos

1. Fazer um programa para receber um número inteiro de segundos do usuário e imprimir a quantidade correspondente em horas, minutos e segundos.

2. Fazer um programa para receber 3 valores inteiros do usuário e mostrar a sua média (que pode não ser inteira).

Aula 2 – Estruturas condicionais

3. Fazer um programa para ler um número do usuário e determinar se este número é par ou não par.

4. Fazer um programa para receber valores inteiros X, Y e Z do usuário e determinar se estes valores podem formar os lados de um triângulo. Em caso afirmativo, informar se o triângulo é equilátero, isósceles ou escaleno.

5. Fazer um programa que recebe 3 valores não inteiros do usuário e mostra o maior deles, o menor deles e a média.

6. Fazer um programa que recebe um símbolo de operação do usuário (+, -, / ou *) e dois números reais. O programa deve retornar o resultado da operação recebida sobre estes dois números.

Aula 3- Estruturas de repetição

7. O número 3025 possui a seguinte característica: 30 + 25 = 55 -> 55*55 = 3025. Fazer um programa para obter todos os números de 4 algarismos com a mesma característica do número 3025.

8. Fazer um programa para mostrar os 100 primeiros termos da série de Fibonacci.

9. Fazer um programa para mostrar todos os números perfeitos entre 1 e 100.

10. Fazer um programa para receber um número inteiro do usuário e determinar se este número é primo ou não.

11. Fazer um programa para receber um número do usuário e decompô-lo em fatores primos.

12. Fazer um programa para receber dois números do usuário e calcular o seu MDC utilizando o método de Euclides. O programa deve continuar pedindo dois números até que 0 e 0 sejam fornecidos.

13. Fazer um programa para receber dois números inteiros do usuário e mostrar o seu MMC (mínimo múltiplo comum).

14. Dada a afirmação: “A tem o dobro da idade que B tinha quando A tinha a idade que B tem. Quando B tiver a idade de A, somarão 81 anos.”. Fazer um programa para calcular as idades de A e B no método “força bruta”.

15. Fazer um programa para medir os reflexos do usuário. O programa deve:
    a. Mostrar a palavra “Saque” após um tempo aleatório
    b. Contar o tempo (em qualquer unidade) até que o usuário digite uma tecla e mostrar esse tempo.
    c. Dicas: random() e kbhit().

16. Fazer um programa para mostrar a soma de todos os números 4 do dominó.

17. Fazer um programa no qual o usuário vai entrando sucessivamente com valores positivos. Quando o usuário entrar com um valor negativo o programa pára de pedir valores e calcula a média dos valores já fornecidos.

18. Fazer um programa para receber dois números do tipo *unsigned int* do usuário e determinar se um número é permutação do outro ou não. Ex: 431 é permutação de 143, 42 é permutação de 204, 1211 é permutação de 1112, etc.

19. Fazer um programa que sorteie um número de 0 a 100 e que permita que o usuário (sem conhecer o número sorteado) tente acertar. Caso não acerte, o programa deve imprimir uma mensagem informando se o número sorteado é maior ou menor que a tentativa feita. Ao acertar o número, o programa deve imprimir a quantidade de tentativas feitas.

20. Escreva um programa que calcule o salário semanal de um trabalhador. As entradas são o número de horas trabalhadas na semana e o valor da hora. Até 40 h/semana não se acrescenta nenhum adicional. Acima de 40h e até 60h há um bônus de 50% para essas horas. Acima de 60h há um bônus de 100% para essas horas.

21. Fazer um programa para encontrar todos os pares de números amigáveis entre 1 e 100000. Um par de números é amigável quando cada um deles é igual à soma dos divisores do outro.

22. Faça um programa que sorteie um número aleatório entre 0 e 500 e pergunte ao usuário qual é o "número mágico". O programa deverá indicar se a tentativa efetuada pelo usuário é maior ou menor que o número mágico e contar o número de tentativas. Quando o usuário conseguir acertar o número o programa deverá classificar o usuário como:

    a. De 1 a 3 tentativas: muito sortudo

    b. De 4 a 6 tentativas: sortudo

c. De 7 a 10 tentativas: normal

d. > 10 tentativas: tente novamente

23. Faça um programa que receba do usuário o número de lados e o tamanho dos lados de um polígono regular e imprima o valor da área do polígono. O programa deve utilizar uma estrutura *switch-case* para decidir que fórmula de cálculo utilizar, de acordo com o número de lados do polígono. Se o número de lados for diferente de 3, 4 ou 6 o programa deve informar: "não sei calcular a área". Áreas:

    a. Triângulo: A = L*L*1.73/4

    b. Quadrado: A = L*L

    c. Hexágono: A = 6*L*L*1.73/4;

24. Um pecuarista possui uma determinada quantia de bois, que possuem um identificador numérico (de 1 a n) cada um. Faça um programa que:
    a. receba o peso de cada boi, um por vez, e o armazene em um vetor. Se o peso digitado for 0 significa que não há mais bois a serem digitados;
    b. mostre a lista de todos os bois com seus identificadores e também os identificadores do boi mais gordo e do boi mais magro. Se houver dois ou mais bois mais gordos ou mais magros mostrar o de menor identificador;
    c. Faça o mesmo programa considerando que o número de bois é fixo e igual a dez.

25. Escrever um programa para ler um número inteiro do usuário e exibir o maior número primo que seja menor do que o número digitado.

26. Fazer um programa para exibir os n primeiros múltiplos simultâneos de dois números dados.

Aula 4 – Funções

27. Implementar a função RAIZQUADRADA. Esta função deve:
    a. Receber um número do tipo *float* como parâmetro.
    b. Retornar a raiz quadrada do número recebido, de tal maneira que esta raiz, quando elevada ao quadrado, apresente um erro máximo de 0.01% em relação ao valor do parâmetro.

28. Implementar a função INVERTE que recebe um número *unsigned int* como parâmetro e retorna este número escrito ao contrário. Ex: 431 <-> 134.

29. Implementar a função *doublé POWER (double base, doublé expoente)*, que retorna o valor de *base* elevado a *expoente*. Dicas:
    a. Transformar o expoente em uma razão de inteiros (multiplicando ambos por 10 até o numerador ficar inteiro).

b. Simplificar a razão de inteiros com sucessivas divisões de numerador e denominador.
c. Calcular $base^{numerador}$
d. Calcular $(base^{numerador})^{1/denominador}$. Utilizar a função de raiz anteriormente implementada.
e. Se $base^{numerador}$ estourar a faixa dos *doublé*, dividir *numerador* e *denominador* por 10 e repetir.
f. Se *numerador* for negativo, resultado é 1/resultado.

30. Fazer uma função que recebe um mês e um ano como parâmetros e retorna o número de dias daquele mês daquele ano. Dica: um ano é bissexto quando é múltiplo de 4 e não múltiplo de 100, ou também quando é múltiplo de 400.

31. Faça uma função que recebe, por parâmetro, a hora de inicio e a hora de término de um jogo, ambas subdivididas em 2 valores distintos: horas e minutos. A função deve retornar, a duração do jogo em minutos, considerando que o tempo máximo de duração de um jogo é de 24 horas e que o jogo pode começar em um dia e terminar no outro.

Aula 5 – Recursividade

32. Escreva a função para cálculo do N-ésimo termo da série de Fibonacci utilizando recursividade.

33. Implementar a função EXP com as seguintes características:
    a. Recebe um valor de *base* e um valor de *expoente* como parâmetros do tipo *unsigned int*.
    b. Recebe o endereço de uma variável, do tipo *unsigned int*, em cujo conteúdo será armazenado o resultado de $base^{expoente}$.
    c. Retorna 1 se foi bem sucedida e 0 se a exponenciação não pôde ser calculada devido a estouro.
    d. Utiliza a característica recursiva da exponenciação:
        i. $Base^{exp} = base.base^{exp-1}$, $exp > 0$
        ii. $Base^{exp} = 1$, $exp == 0$

34. Considere uma partida de futebol entre duas equipes A x B, cujo placar final é m x n, em que *m* e *n* são números de gols marcados por A e B, respectivamente. Escreva um algoritmo recursivo que imprima todas as possíveis sucessões de gols marcados. Por exemplo, para um placar final de 3 x 1, as possíveis sucessões de gols são “AAAB”, “AABA”, “ABAA” e “BAAA”.

35. Torre de Hanói: considerando 3 torres, o objetivo é transferir 3 discos que estão na torre A para a torre C, usando uma torre B como auxiliar. Somente o último disco de cima de uma pilha pode ser deslocado para outra, e um disco maior nunca pode ser colocado sobre um menor. Implementar uma função recursiva que mostra a seqüência de movimentos para resolver o problema da Torre de Hanói.

Aula 6 – Ponteiros

36. Exercício 33.

37. Fazer uma função FATORES que:
    a. Recebe 3 parâmetros: um vetor de inteiros, um número inteiro *n* passado por valor e outro número *x* passado como ponteiro.
    b. Retorna um número inteiro.
    c. Decompõe o número *n* em fatores primos e armazena-os nas posições do vetor. O conteúdo de *x* deve receber o número de fatores primos encontrados. Caso o número de fatores encontrados seja maior que 10, a função deve retornar 1, do contrário deve retornar 0.

38. Fazer uma função para:
    a. Receber dois ponteiros para char (char*) como parâmetro e um número representando uma certa quantidade de caracteres.
    b. procurar, no vetor apontado pelo parâmetro 1, o primeiro caracter de espaço (‘ ‘) ou o fim de vetor (representado pela quantidade fornecida no parâmetro 3).
    c. copiar os caracteres anteriores ao espaço no vetor indicado pelo segundo parâmetro.
    d. retornar o número de caracteres copiados.

39. Fazer um programa para:
    a. Receber uma frase do usuário, caracter a caracter usando getch() e armazenando no vetor (máx. 30 caracteres). Quando o usuário digita **enter** (‘\r’) a recepção é finalizada.
    b. mostrar cada palavra da frase em uma linha separada, utilizando a função do exercício 3.

40. Fazer um programa para:
    a. declarar variáveis a, b, c, d do tipo int.
    b. declarar variáveis e, f, g, h do tipo float.
    c. declarar vetor v de 10 elementos do tipo char.
    d. declarar variável x do tipo int.
    e. criar um ponteiro apontando para o endereço de a.
    f. incrementar o ponteiro, mostrando o conteúdo do endereço apontado (em forma de número). Caso o endereço coincida com o endereço de alguma outra variável, informar o fato.

41. Fazer uma função com as seguintes características:
    a. recebe dois números inteiros do usuário.
    b. retorna 1 se os números são iguais, 0 se são diferentes e –1 se a soma ou o produto estoura a faixa dos inteiros. Além disso, retorna a soma e o produto dos dois números.
    c. Fazer um programa para receber dois números do usuário, chamar a função e mostrar se os números são iguais. Além disso, mostrar sua soma e seu produto.

42. Fazer uma função que:

a. receba 3 números como parâmetros: A, B e C.
b. ordene de tal forma que, ao final da função, A contenha o menor número e C o maior.
c. Fazer um programa que receba 3 números do usuário, chame a função e mostre os números ordenados.

43. Escreva uma função CALCULA que:
    a. receba como parâmetros duas variáveis inteiras, X e Y;
    b. retorne em X a soma de X e Y;
    c. retorne em Y a subtração de X e Y.

    Pergunta: a passagem dos parâmetros para a função deve ser por valor ou por referência?

44. Fazer uma função DIVS que:
    a. recebe como parâmetro um número inteiro n por valor e dois números inteiros max e min por referência;
    b. retorna 0 se o número num é primo e 1 caso contrário. Se o número não for primo, as variáveis max e min devem assumir os valores do menor e do maior divisores inteiros do número, respectivamente, desconsiderando o número 1 e o próprio número num.

Aula 7 – Vetores

45. Faça um programa que dado o vetor unidimensional [2; 4; 35; 50; 23; 17; 9; 12; 27; 5] retorne:
    a. maior valor
    b. média dos valores
    c. os valores dispostos em ordem crescente
    d. sub conjunto de valores primos que está contido no vetor

46. Faça um programa que:
    a. leia 7 valores inteiros e os armazene em um vetor. Listar o vetor com as referidas posições de armazenamento de cada valor.
    b. ofereça uma função de pesquisa onde dado um valor inteiro qualquer de entrada retornar a posição deste valor dentro do vetor, e caso este valor não esteja presente no vetor retornar –1.
    c. ofereça uma função que troque os valores contido no vetor pela seguinte política: cada elemento i dentro do vetor será substituído pela soma de todos os (i-1) elementos mais o elemento i. Por exemplo, dado um vetor [1; 2; 3; 4; 5] após a aplicação da função teríamos esse vetor preenchido com os seguintes valores [1; 3; 6; 10; 15]. Para esta tarefa utilize um vetor auxiliar.

47. Faça um programa que, dados dois vetores bidimensionais (matrizes A e B) com dimensões de no máximo 5x5 elementos, retorne:
    a. a soma destas duas matrizes

b. a soma das diagonais de cada matriz
c. a multiplicação das duas matrizes

48. Faça um programa para:
    a. receber as dimensões M e N da matriz A (M e N <= 5)
    b. receber os MxN elementos da matriz A
    c. receber as dimensões J e K da matriz B (J e K <=5, J = N)
    d. receber os JxK elementos da matriz B
    e. calcular e mostrar a matriz C, de dimensões MxK, que é o produto das matrizes A e B.

49. Faça um programa para receber do usuário a dimensão de um vetor (máx. 20), os elementos desse vetor e efetuar a sua ordenação utilizando o método da bolha (*bubble-sort*).

50. Vamos supor que várias pedras do jogo de xadrez estão no tabuleiro. Para facilitar a indicação das peças, vamos convencionar:
    1 – peões 3 – torres 5 – reis 0 – ausência de peças
    2 – cavalos 4 – bispos 6 – rainhas

    O tabuleiro é o seguinte:

1	3	0	5	4	0	2	1
1	0	1	0	0	1	0	0
0	0	0	0	1	0	6	0
1	0	0	1	1	0	0	1
0	1	0	4	0	0	1	0
0	0	3	1	0	0	1	1
1	0	6	6	0	0	1	0
1	0	5	0	1	1	0	6

    a) Construa um programa que determine a soma total entre peões ou bispos e a quantidade de posições com ausência de peças;
    b) Escreva outro programa que determine qual a quantidade de cada tipo de peça no tabuleiro.

51. A distância entre várias cidades é dada pela tabela abaixo (em km):

	1	2	3	4	5
1	00	15	30	05	12
2	15	00	10	17	28
3	30	10	00	03	11
4	05	17	03	00	80
5	12	28	11	80	00

    a) Construa um programa que leia a tabela acima e informe ao usuário a distância entre duas cidades por ele requisitadas, até que ele entre com o código 0 para ambas as cidades;
    b) Elabore um programa que imprima a tabela sem repetições, isto é, se a distância entre as cidades 1 e 3 foi emitida, não é necessário emitir a distância entre 3 e 1;

c) Dado um determinado percurso, imprima o total percorrido:
Exemplo: dado o percurso 1, 2, 3, 2, 5, 1, 4, teremos:
15 + 10 + 10 + 28 + 12 + 5 = 80 km.

52. Um cinema que possui capacidade de 20 lugares está sempre lotado. Certo dia cada espectador respondeu a um questionário, onde constava:
- sua idade;
- sua opinião em relação ao filme, que podia ser: **ótimo, bom, regular, ruim** ou **péssimo.**

Elabore um programa que, recebendo estes dados calcule e mostre:

a. a quantidade de respostas ótimo;
b. a diferença percentual entre respostas bom e regular;
c. a média de idade das pessoas que responderam ruim;
d. a porcentagem de respostas péssimo e a maior idade que utilizou esta opção;
e. a diferença de idade entre a maior idade que respondeu ótimo e a maior idade que respondeu ruim.

53. Faça um programa que receba uma matriz 5x5 valores do tipo int do usuário, um valor de cada vez, e imprima a sua matriz transposta (Obs: a matriz transposta é obtida permutando-se as linhas e as colunas de uma matriz.

54. Escreva um programa que leia uma matriz n x m do usuário e a transforme em um vetor unidimensional de n.m posições

55. Fazer um programa para:

a. receber 3 notas parciais do aluno em um vetor e a nota do exame em uma variável separada (-1 se o aluno não fez exame).
b. chamar a função SITUACAO, com as seguintes características:
    i. Parâmetros: vetor de notas parciais e nota do exame
    ii. Retorno: 0 se o aluno está reprovado direto, 1 se o aluno está reprovado em exame, 2 se o aluno está aprovado em exame e 3 se ele está aprovado direto.

Aula 8 – Strings

56. Faça um programa que dado um nome completo, retorne a abreviatura deste nome. Não se devem abreviar as preposições como: do, de, etc. A abreviatura deve vir separada por pontos. Ex: Paulo Jose de Almeida Prado. Abreviatura: P.J.A.P.

57. Faça um programa que dado 2 palavras, determine:
    a. Se as palavras são iguais;

b. Caso as palavras sejam diferentes, qual delas tem maior comprimento (não esquecer a possibilidade de existirem palavras diferentes de mesmo tamanho);
c. Verifique se a segunda palavra é uma sub string da primeira:

Exemplo: Palavra 1= **casa**mento
Palavra 2 = **casa**

58. Faça um programa onde o usuário digita 3 informações a respeito de uma pessoa: Nome, endereço e telefone. Concatene essas três informações em uma única string e faça uma contagem de quantas letras do alfabeto estão presentes nesta string (considerando as redundâncias) e também de dígitos numéricos. Os espaços e os caracteres de pontuação devem ser ignorados(as funções de contagem já fazem isso).
Dica: use as funções int isalpha(char cr) e int isdigit(char cr).

Exemplo:
Nome: Ana Claudia
Endereço: Rui Barbosa, 234
Tel: 234-0912

Resultado:
Quantidade de letras pertencentes ao alfabeto = 20.
Quantidade de dígitos numéricos = 10

59. Fazer um programa para:
    a. Receber uma string de no máximo 100 caracteres
    b. Receber uma segunda string e contar quantas vezes a segunda string ocorre dentro da primeira.

60. Fazer um programa para:
    a. Receber uma string do usuário.
    b. Contar quantos ditongos ou hiatos existem na string
    c. Contar quantas duplas de letras repetidas existem na string.

61. Fazer um programa para cadastro e diálogo de login. O programa deve:

    a. Cadastrar um nome de usuário via teclado. O nome de usuário tem, no máximo, 8 caracteres, sendo válidos somente os caracteres numéricos e as letras maiúsculas ou minúsculas. Somente os caracteres válidos devem ser exibidos no console durante a digitação do nome de usuário.
    b. Cadastrar uma senha do usuário via teclado. Esta segue as mesmas regras do nome de usuário, com a diferença de que são exibidos somente asteriscos no console à medida que a senha é digitada.
    c. Receber um novo nome de usuário e uma nova senha, utilizando os mesmos procedimentos descritos nos itens a e b.

d. Comparar o nome de usuário cadastrado com o recebido posteriormente e a senha cadastrada com a senha recebida. Caso sejam idênticos, informar “OK”, do contrário informar “Acesso negado”.

62. Elabore um programa que, dado 2 vetores inteiros de 20 posições, efetue as respectivas operações indicadas por um terceiro vetor de caracteres de 20 posições também fornecido pelo usuário, contendo as quatro operações aritméticas em qualquer combinação, armazenando os resultados num quarto vetor.

63. Elaborar um programa em C que leia uma frase e armazene-a em um vetor de caracteres (cuidado com a leitura!). Depois crie uma função para contar o número de espaços em branco na frase, outra para contar o número de vogais, e outra para contar o número de consoantes.

64. Com o vetor do exercício 8, faça uma função que transfira as consoantes para um vetor e as vogais para outro. Depois mostre cada um dos vetores.

65. Escreva um programa que utilize uma função "replace" que aceita um string como parâmetro e retorna um inteiro. A função substitui todos os espaços do seu parâmetro pelo caracter '-', e retorna o número de substituições feitas. O programa que a usa deverá testar a sua funcionalidade.

66. Escreva um programa que leia texto do teclado, linha a linha, até chegar ao fim de texto (Ctrl-D ou Ctrl-Z). O programa deverá escrever uma estatística do texto lido: nº de palavras, número de linhas em branco, nº total de linhas, nº de letras. O programa deverá usar funções separadas para cada uma das suas tarefas.

67. Fazer um programa que receba uma string de no máximo 20 caracteres do usuário e mostre o conteúdo desta string de forma invertida.

68. Faça um programa que receba uma string do usuário (máx. 20 caracteres) e um caracter qualquer. O programa deve remover todas as ocorrências do caracter da string e mostrar o resultado.

69. Um dos sistemas de encriptação mais antigos é atribuído a Júlio César: se uma letra a ser encriptada é a letra de número N do alfabeto, substitua-a com a letra (N+K), onde K é um número inteiro constante (César utilizava K = 3). Usualmente consideramos o espaço como zero e todos os cálculos são realizados com módulo-27. Dessa forma, para K = 1 a mensagem “Ataque ao amanhecer” se torna “bubrfabpabnboifdfs”. Faça um programa que receba como entrada uma mensagem e um valor de J e retorne a mensagem criptografada pelo código de César. Fraquezas: apenas 26 chaves possíveis. É possível utilizar conhecimento da linguagem para facilitar a busca.

70. Faça um programa que receba como entradas uma lista de nomes em ordem aleatória e ordene essa lista em ordem alfabética.

71. Faça um programa que inverta a ordem das letras de uma string. Utilize o código do exercício 14 para incrementar o código de César adicionando a inversão da mensagem encriptada.

72. Para evitar fraudes, uma máquina de preenchimento de cheques deve preencher as dezenas não utilizadas no valor numérico com asteriscos. Considerando que na loja X não são aceitos cheques de valores maiores que R$ 10.000,00, faça um programa que imprimia na tela o valor numérico do cheque e seu valor por extenso.

73. Escrever uma função que:

    a. receba dois strings como parâmetro, bem como um valor inteiro representando uma posição.

    b. insira o segundo string no primeiro, na posição indicada pelo valor.

    c. Fazer um programa que receba dois strings do usuário, o valor da posição, chame a função anteriormente implementada e exiba o resultado ao usuário.

74. Fazer um programa para receber uma string do usuário (máx. 50 caracteres) e fazer uma estatística dos caracteres digitados. Por exemplo, para a string "O EXERCICIO E FACIL"”, a estatística mostrada será 'O' = 2, ' '=3, 'E' = 3, 'X' = 1, 'R' = 1, 'C' = 3, 'I' = 3, 'F' = 1, 'A' = 1, 'L' = 1

Aula 9 – Estruturas de dados

75. Fazer um programa que receba três nomes de no máximo 15 caracteres cada um (nomes com mais de 15 caracteres devem ser rejeitados) e as idades das respectivas pessoas em um vetor de estruturas de dados. Após o recebimento, listar os 3 nomes e idades que nela foram armazenados.

76. Fazer um programa de diálogo de login semelhante ao exercício 6 de strings, com a diferença de que é possível cadastrar no máximo 10 nomes de usuário e suas respectivas senhas (nomes de usuário repetidos devem ser descartados). No diálogo de login, o programa deve testar se o usuário fornecido existe e se a sua senha confere.

Aula 10 – Alocação Dinâmica

77. Fazer um programa que receba do usuário a quantidade N de números a ser digitada. Em seguida, o programa deve alocar dinamicamente um vetor de N inteiros, receber N números do usuário e armazenar no vetor, e mostrar o maior valor do vetor, o menor valor do vetor e a média dos valores.

Aula 11 – Arquivos

78. Implementar um programa de gerenciamento de high scores. O programa deve:
    a. carregar os high scores de um arquivo e mostrar. Cada high score é composto de um nome (max. 10 caracteres) e um inteiro (pontuação)
    b. pedir ao usuário o seu nome e a sua pontuação no jogo.
    c. posicionar os dados do usuário na tabela de highest scores (max. 5) e regravar no arquivo.