Exercícios de Linguagem C

Aula 1 – Aspectos básicos

1. Fazer um programa para
   a. receber do usuário um tempo em segundos, correspondente à duração de um evento qualquer (por ex. jogo de futebol)
   b. calcular e mostrar ao usuário o tempo equivalente 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).

3. Fazer um programa para calcular uma trajetória parabólica. O programa deve:

   a. Receber do usuário a altura relativa ao solo de onde o projétil é lançado, a velocidade inicial em m/s e o ângulo de lançamento
   b. Calcular e mostrar a quantos metros de distância o projétil atingirá o chão, considerando aceleração da gravidade igual a 9,81 m/s^2.

Aula 2 – Estruturas condicionais

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

5. 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.

6. Fazer um programa que recebe 3 valores não inteiros do usuário e mostra o maior deles, o menor deles e o valor intermediário.

7. 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.

8. Fazer um programa que recebe duas notas parciais de um aluno (0 a 100) e informa se ele está aprovado (média maior ou igual a 70), em exame final (média entre 40 e 69) ou reprovado (média menor do que 40). Caso esteja em exame o programa deve pedir a nota do exame (0 a 100) e informar se o aluno está aprovado (média mais nota do exame maior ou igual a 100) ou reprovado (média mais nota do exame menor do que 100).

9. Fazer um programa para solicitar ao usuário um número entre 0 e 99 e mostrar este número por extenso. Se o usuário introduzir um número que não está neste intervalo, mostre: "número inválido".
   Fonte:http://www.bernhard.pro.br/disciplinas/java/ensino/java-L01.pdf

10. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Escreva um programa que receba a velocidade máxima em uma avenida e a velocidade com que um motorista estava dirigindo nela. Calcule a multa que o motorista vai receber, considerando que são pagos R$ 5,00 por cada km/h que estiver acima da velocidade permitida (considere apenas números inteiros). Se a velocidade do motorista estiver dentro do limite, o programa deve informar que não há multa.

11. Escreva um programa para receber do usuário um mês (de 1 a 12) e um ano (de 0 D.C. em diante) e mostrar o número de dias daquele mês daquele ano. Utilizar *switch* e levar em consideração anos bissextos para o cálculo.

12. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Suponha que você está programando um módulo contador de cédulas para caixas eletrônicos. Escreva um programa que informa com quantas cédulas de Real podemos representar um dado valor. Exemplo de resposta: R$ 218 = 2 cédulas de 100, 1 cédula de 10, 1 cédula de 5, 1 cédula de 2 e 1 cédula de 1. Procure minimizar o número de cédulas usadas. Desconsidere valores com centavos, e suponha que a máquina sempre tem disponíveis as cédulas necessárias.

13. 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.

14. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Zeca está organizando um bolão de futebol. Segundo suas regras, os apostadores informam o placar do jogo e ganham 10 pontos se acertarem o vencedor ou se acertarem que foi empate, e ganham mais 5 pontos para o placar de cada time que acertarem. A tabela a seguir dá um exemplo, considerando que o placar real foi 3x2:

Placar apostado	Pontos	Razão
0x1	0	Não acertou o vencedor e nem o número de gols dos times.
0x2	5	Não acertou o vencedor, mas acertou o número de gols do segundo time.
3x5	5	Não acertou o vencedor, mas acertou o número de gols do primeiro time.
1x0	10	Acertou o vencedor, mas não acertou o número de gols dos times.
3x1	15	Acertou o vencedor e o número de gols do primeiro time.
3x2	20	Acertou o vencedor e o número de gols de ambos os times.

Escreva um programa que requisita do usuário o placar apostado e depois o placar do jogo e informa quantos pontos o apostador fez.

15. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) A tabela abaixo foi copiada do website da Receita Federal, e traz as alíquotas do imposto de renda de pessoa física retido na fonte para o exercício de 2013:

Base de cálculo mensal em R$	Alíquota %	Parcela a deduzir do imposto em R$
Até 1.637,11	-	-
De 1.637,12 até 2.453,50	7,5	122,78
De 2.453,51 até 3.271,38	15,0	306,80
De 3.271,39 até 4.087,65	22,5	552,15
Acima de 4.087,65	27,5	756,53

A base de cálculo é dada pelo salário mensal, com certas deduções, como dependentes e contribuição previdenciária. Abstraindo estes e outros detalhes, o imposto devido é calculado tomando a base de cálculo mensal, verificando a faixa na qual ela se encontra, aplicando a alíquota correspondente, e reduzindo o valor final da parcela a deduzir. Por exemplo, se a base de cálculo é de R$10.000,00, a alíquota é de 27,5%, ou seja, R$2750,00. Deduzimos R$756,53 da parcela e obtemos como resultado final R$1993,47 de imposto devido.

Escreva um programa que receba como entrada a base de cálculo mensal de um trabalhador e retorne o imposto de renda devido.

16. 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;

Aula 3 - Estruturas de repetição

17. 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.

18. Fazer um programa para mostrar os 15 primeiros termos da série de Fibonacci.

19. Fazer um programa para mostrar todos os números perfeitos entre 1 e 100. Número perfeito é todo número inteiro cuja soma dos seus divisores é igual ao dobro do próprio número.

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

21. Fazer um programa para simular a trajetória parabólica de um projétil (como, por exemplo, no jogo *Angry Birds*) arremessado de uma altura *h*, com velocidade inicial *V* e ângulo de

lançamento *ang.* Considerar que a posição *(x,y)* do projétil deve ser atualizada e mostrada via *printf* em um intervalo de tempo correspondente a *dt* (por exemplo, 10 ms).

22. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Escreva um programa que calcule o quociente e o resto da divisão de dois números inteiros dados, usando apenas as operações de soma e/ou subtração.

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

24. 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.

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

26. 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" (tentativa e erro com todos os valores inteiros possíveis).

27. Fazer um programa para medir os reflexos do usuário. O programa deve:
    a. Mostrar a palavra "Agora!" após um tempo aleatório e um número, também aleatório
    b. Medir o tempo até que o usuário digite o número pedido e mostrar esse tempo.

    Dica: usar a função *clock* da biblioteca *time.h* (verificar exemplos na internet ou pedir ao professor),

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

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

30. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Escreva um programa que lê um inteiro positivo e verifica se ele contém o dígito 3 em qualquer posição.

31. Como generalização do exercício anterior, escreva um programa que lê dois inteiros positivos *n1* e *n2* e verifica se *n1* contém a sequência de dígitos de *n2* em qualquer posição. Por exemplo, isto ocorreria para *n1 = 3623* e *n2 = 62,* ou para *n1 = 3623* e *n2 = 36.*

32. 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. O programa só deve aceitar números positivos.

33. 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.

34. 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

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

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

37. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Escreva um programa no qual o usuário digita uma quantidade arbitrária de números inteiros positivos. Quando o usuário digitar um número menor ou igual a 0, o programa deve indicar o tamanho da maior sequência crescente observada. Por exemplo, se os números digitados forem 5, 10, 3, 2, 4, 7, 9, 8, 5, a maior sequência crescente é 2, 4, 7, 9, então o programa deve mostrar na tela que a maior sequência crescente tem 4 números. Já a sequência 10, 8, 7, 5, 2 está em ordem decrescente, portanto a maior sequência crescente observada tem tamanho 1.

38. Chico tem 1,50 metro e cresce 2 centímetros por ano, enquanto Zé tem 1,10 metro e cresce 3 centímetros por ano. Construa um programa que calcule e imprima quantos anos serão necessários para que Zé seja maior que Chico.

39. Um passageiro corre à sua velocidade máxima de *8 m/s* para pegar um trem. Quando está à distância *d* da porta de entrada, o trem principia a rodar com aceleração constante *a = 1 m/(s*s)*, afastando-se.
    a. Se *d = 30 m* e se o passageiro continua a correr, conseguirá ou não pegar o trem? Responda a essa pergunta elaborando um programa em C que simula os deslocamentos do passageiro e do trem, com intervalos de tempo variando entre *1 s e 0,1 s*. Avalie os resultados obtidos frente ao resultado analítico (obtido através da resolução da equação).
    b. A distância crítica de separação inicial é chamada de *dc*. Adaptar o programa anterior para variar a distância inicial *d* e obter a distância crítica por comparação. Com a distância crítica de separação *dc*, qual a velocidade do trem quando o passageiro consegue pegá-lo? Qual é a velocidade média do trem no intervalo de tempo *t = 0* até este instante? Qual é o valor de *dc*?
    c. Fazer um programa em C para desenhar a função posição *x(t)* do trem, com *x = 0* em *t = 0*. No mesmo gráfico, desenhar a função *x(t)* do passageiro, com diversas distâncias de separação inicial *d*, incluindo a distância *d = 30 m* e a distância crítica de separação *dc* que lhe permite pegar o trem por um átimo.

40. Um corpo se move sobre uma reta e duplica sua velocidade, a cada segundo, durante os primeiros 10s. Seja *2 m/s* a velocidade inicial. Qual é a velocidade média nos primeiros *10s*? Responda a essa pergunta elaborando um programa em C que simula a variação da velocidade e o deslocamento do corpo, com intervalos de tempo variando entre *1 s e 0,1 s.*

41. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Um agricultor possui 1 (uma) espiga de milho. Cada espiga tem 150 grãos, e cada grão pesa 1g (um grama). Escreva um programa para determinar quantos anos serão necessários para que o agricultor colha mais de cem toneladas de milho (1T = 1000Kg, 1Kg =1000g), sendo que:
    a. A cada ano ele planta todos os grãos da colheita anterior.

a. Há apenas uma colheita por ano
b.10% (dez por cento) dos grãos não germinam (morrem sem produzir).
c. Cada grão que germina produz duas espigas de milho.

Considere que a quantidade de terra disponível é sempre suficiente para o plantio.

42. A nota final de Fund Prog I é calculada conforme a fórmula fornecida pelo professor no primeiro dia de aula (ver Moodle). Após a primeira prova, primeira lista de exercícios e a primeira entrega do trabalho, os alunos têm muito interesse em saber de que nota precisam nas demais avaliações para serem aprovados. Faça um programa que:
    a. Receba do usuário os valores das notas da primeira prova, primeira lista e se a proposta de trabalho foi entregue atrasada ou não (propostas em atraso geram uma penalidade de 20% na nota do projeto final).
    b. Calcule e mostre todas as combinações possíveis de notas das demais avaliações (segunda prova, segunda lista e nota final do trabalho) que permitem que ele seja aprovado. Considerar que a resolução (variação mínima) para cada uma das notas é de 1 ponto em 100.

Aula 4 – Funções

43. Fazer uma função *situacao_aluno* que:
    a. recebe como parâmetros: média final do aluno (0 a 100), número de faltas e quantidade de horas-aula no semestre
    b. retorna 1 se o aluno foi aprovado, de acordo com os critérios da UTFPR, e 0 caso contrário.

    Critérios: média final maior ou igual a 60 e frequência maior ou igual a 75%

    Fazer o teste da chamada desta função no *main*, recebendo valores relativos a um aluno e informando se ele foi aprovado ou não.

44. 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.
    c. **Não utilize outras funções prontas (sqrt, pow) – deve implementar utilizando um método numérico.**

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

46. Implementar a função *double power (double base, double 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 *double*, dividir *numerador* e *denominador* por 10 e repetir.
f. Se *numerador* for negativo, resultado é 1/resultado.

47. 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.

48. Faça uma função que recebe, por parâmetros, 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.

49. (*Cortesia Prof. Bogdan Tomoyuki Nassu*) Escreva uma função que arredonda um valor dado. O número deve ser arredondado para o inteiro mais próximo. Se o número for equidistante de dois inteiros, deve ser arredondado para o valor de maior magnitude. Ou seja, 1.5 é arredondado para 2, e -1.5 é arredondado para -2. O protótipo da função deve ser:

```
int arredonda (double x);
```

50. (*Cortesia Prof. Bogdan Tomoyuki Nassu*) Implementar a função cujo cabeçalho é apresentado a seguir:

```
unsigned char calculaParidade (unsigned char b);
```

Interferências, ruídos e outros fenômenos que prejudicam a integridade dos dados são problemas fundamentais quando computadores se comunicam em rede. Para detectar alterações em bits, os dados são sempre enviados com redundâncias computadas a partir dos bits originais. Este tipo de técnica de detecção de erros costuma receber o nome de checksum, e segue o mesmo princípio dos dígitos verificadores presentes em diversos documentos e identificadores numéricos (por exemplo, números de contas e agências bancárias).

Uma das técnicas de detecção de erros mais simples e mais usadas é o teste de paridade. Cada byte é enviado junto com um bit adicional, que indica se o número de bits com valor 1 no byte é par (bit redundante = 0) ou ímpar (bit redundante = 1). Por exemplo um byte com o valor 8 tem os bits 00001000, ou seja, apenas 1 bit "setado", portanto a sua paridade é 1. Já um byte com o valor 0x55 é representado pelos bits 01010101 – 4 bits "setados", portanto a sua paridade é 0.

A função *calculaParidade* deve receber como parâmetro um byte enviado ou recebido através de uma conexão, e retornar o valor do bit redundante (0 ou 1).

51. (*Cortesia Prof. Bogdan Tomoyuki Nassu*) Implementar a função cujo cabeçalho é apresentado a seguir:

```
float calculaValorDevido (float peso, float custo_fixo, float preco_kg,
float largura_faixa, float desconto_faixa, float desconto_max);
```

Gisele acaba de comprar um restaurante de comida por quilo que vinha dando prejuízos ao proprietário anterior. Gisele tem uma série de ideias que pretende implantar para que o empreendimento se torne lucrativo. As ideias envolvem mudanças na decoração, equipe, cardápio, público-alvo e política de preços. Uma das causas do prejuízo, notou Gisele, é que o custo da estrutura para atender a um cliente não é proporcional ao seu consumo. Por exemplo, o salário de um cozinheiro é fixo, e o esforço necessário para preparar a comida consumida por um cliente não dobra quando o consumo dobra. Além disso, existem custos fixos, como o aluguel do ponto, impostos municipais, e as despesas com limpeza. Com tudo isso, pode-se afirmar que um cliente que consome 200g é menos lucrativo para o restaurante do que outro que consome 500g não apenas em números absolutos, mas também proporcionalmente.

Para amenizar o problema, Gisele resolveu adotar uma política de "descontos progressivos". Funciona assim: todo cliente deve pagar um valor fixo pelo atendimento. Sobre este valor, adiciona-se um segundo valor, por kg. Os primeiros 100g consumidos são calculados considerando o preço normal. Os próximos 100g recebem um desconto de 10%, os 100g seguintes um desconto de mais 10%, e assim por diante, até um desconto máximo de 50%, para o que for consumido acima de 500g. Esta medida tem como efeito colateral estimular que os clientes consumam mais, pois têm a sensação de estarem pagando cada vez menos pela comida.

A empresa onde você trabalha foi contratada para desenvolver os sistemas usados no restaurante de Gisele. Você ficou encarregado da função que calcula quanto um cliente deve pagar, com base no peso do prato e em certos parâmetros – que não foram fixados para que possam ser reconfigurados com o passar do tempo, ou modificados de acordo com a necessidade do estabelecimento. A função recebe como parâmetros o peso da comida do cliente em gramas (com o peso do prato já descontado), o preço fixo e o preço por kg em Reais, a largura das faixas de desconto em gramas, o bônus no desconto progressivo, e o desconto máximo. O valor de retorno deve ser o valor devido pelo cliente, em Reais, sem arredondamentos (i.e. não se preocupe com valores "quebrados").

Aula 5 – Recursividade

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

53. 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. Retorna o resultado de $base^{expoente}$ na forma de um *unsigned int.*
    c. Armazena o valor 1 na variável global *status* 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. $Base^{exp} = 1$, $exp == 0$

54. 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".

55. 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.

56. Escreva uma função recursiva que calcule a soma dos dígitos de um número inteiro. Por exemplo, se a entrada for 123, a saída deverá ser 1+2+3 = 6.
Fonte: https://programacaodescomplicada.files.wordpress.com/2012/10/lista-recursc3a3o.pdf

57. Crie uma função recursiva que receba um número inteiro positivo N e calcule o somatório dos números de 1 a N.
Fonte: https://programacaodescomplicada.files.wordpress.com/2012/10/lista-recursc3a3o.pdf

58. Escreva uma função recursiva que determine quantas vezes um dígito *K* ocorre em um número natural *N*. Por exemplo, o dígito 2 ocorre 3 vezes em 762021192.
Fonte: https://programacaodescomplicada.files.wordpress.com/2012/10/lista-recursc3a3o.pdf

59. Dado um número *n* na base decimal, escreva uma função recursiva em C que imprime este número na base binária.
Fonte: https://programacaodescomplicada.files.wordpress.com/2012/10/lista-recursc3a3o.pdf

Aula 6 – Ponteiros

60. Fazer uma função *recebe_vetor* com as seguintes características:
    - Parâmetros: ponteiro para ponteiro, que indica a posição inicial de um vetor a ser alocado dinamicamente pela função.
    - Retorno: tamanho alocado (ou 0 se não foi possível).
    - Descrição: recebe do usuário o tamanho do vetor a ser criado (vetor de *int*), cada um dos dados do vetor e retorna o vetor preenchido bem como o seu tamanho.

    A função deve poder ser utilizada com o seguinte código-fonte:

```
int main()
{
      int * vetor;
      int n, i;
      ...
      n = recebe_vetor(&vetor);
```

```
        for (i = 0; i < n; i++)
        {
              /* faz algum processamento sobre vetor[i] */
        }
        ...
}
```

61. 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 até um limite de 10. 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 e o vetor deve receber somente até o décimo fator primo, do contrário deve retornar 0.

62. 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.

63. 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.

64. Fazer uma função com as seguintes características:
    a. recebe dois números inteiros como parâmetros.
    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.

65. 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.

66. 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?

67. Fazer uma função *divs* que:
    a. recebe como parâmetro um número inteiro *num* por valor e dois números inteiros *max* e *min* por referência (ponteiros);
    b. retorna 0 se o número *num* é primo e 1 caso contrário. Se o número não for primo, os conteúdos dos endereços apontados por *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*.

68. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Escreva uma função que recebe como parâmetros um vetor de inteiros positivos, o número de elementos dele, e ponteiros para argumentos de saída, nos quais devem ser armazenados os valores máximo e mínimo do vetor, assim como a média dos valores. O protótipo da função deve ser:

```
void minMaxMedio (int* vetor, int tam, int* min, int* max, double* medio);
```

69. (*Cortesia do Prof. Bogdan Tomoyuki Nassu)* Em jogos de tempo real, o tempo é dividido em game ticks - intervalos de tempo entre um evento e outro. Em jogos como os da série *Street Fighter*, quando o jogador faz um movimento de pulo, o computador calcula a posição do personagem a cada game tick, seguindo uma trajetória parabólica.

a. Escreva uma função com o protótipo abaixo:

```
int jumpPosition (double v0, double theta, double t, double* x, double* y);
```

A função deve calcular a posição do personagem no tempo t, considerando que ele segue uma trajetória parabólica*, com velocidade inicial v0 e ângulo theta, dado em graus. Suponha que a aceleração da gravidade é 9.8m/s2. Os parâmetros x e y são usados como saídas, para indicar a posição do personagem. Esta posição é dada com relação à posição inicial (i.e. considere que a posição inicial é em x=0 e y=0). A função deve retornar 0 se o personagem já tiver retornado ao solo no tempo t, ou 1 do contrário.

*http://en.wikipedia.org/wiki/Projectile_motion

Dica: a biblioteca-padrão da linguagem C tem várias implementações de funções matemáticas.

b. Escreva um programa que chama a função *jumpPosition* sucessivas vezes, para diferentes valores de t, mostrando (em texto!) a posição do personagem do momento em que ele inicia o movimento até retornar ao solo. O game tick deve ser definido como uma constante. Já os valores de v0 e theta passados para a função devem ser informados pelo usuário (na prática, eles seriam diferentes para cada personagem do jogo). Exemplos de entrada e saída:

Dados `v0=30` (m/s), `theta=45` e `GAME_TICK=0.5`.

t	x	y
0.00	0.00	0.00
0.50	10.61	9.38
1.00	21.21	16.31
1.50	31.82	20.79
2.00	42.43	22.83
2.50	53.03	22.41
3.00	63.64	19.54
3.50	74.25	14.22
4.00	84.85	6.45
4.50	95.46	-3.77

70. (*Cortesia do Prof. Bogdan Tomoyuki Nassu)* Em programas que manipulam strings, a leitura de cada string é feita em um buffer com o tamanho máximo para as strings lidas. Por exemplo, o código abaixo lê uma string de até 1024 caracteres (contando o '\0' final):

```
#define BUFLEN 1024
(...)
char buffer [BUFLEN];
fgets (buffer, BUFLEN, stdin);
```

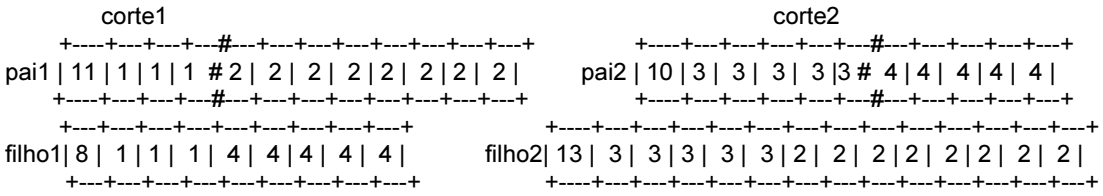
Neste código, o buffer que contém a string possui tamanho 1024, mesmo que a string digitada seja simplesmente "oi". A maior parte do espaço não é usada. Além disso, se for necessário ler outra string usando o mesmo buffer, o conteúdo da primeira string digitada será substituído. Nestas situações, uma forma de economizar memória é usar o buffer maior apenas para ler as strings, mas alocar cada nova string em outro espaço, que tem apenas o tamanho necessário.

Com base nisso, escreva uma função *char* empacotaString (char* string)*, que recebe como parâmetro um buffer contendo uma string e retorna uma cópia da string, mas em um espaço que tem apenas o tamanho necessário. A nova string deve ser alocada dentro da função, mas a responsabilidade de desalocá-la é do chamador.

71. (*Cortesia do Prof. Bogdan Tomoyuki Nassu)* O vazamento de memória é um problema grave, e mesmo programadores experientes podem deixar de perceber a sua ocorrência. Para testar as consequências deste problema, crie um programa com um laço infinito. Dentro do laço, aloque um vetor de 1024 bytes. Para evitar que o programa rode rápido demais e que otimizações do compilador descartem o vetor, use a função *printf* para imprimir o conteúdo da primeira posição do vetor (o conteúdo será "lixo" - neste exemplo, o conteúdo do vetor é irrelevante). Não desaloque este vetor. Isso fará com que cada iteração do laço crie um novo vetor. Acompanhe a utilização de memória usando as ferramentas do sistema operacional.

No Windows, pressione CTRL+ALT+DEL, escolha a opção de iniciar o Gerenciador de Tarefas, e observe na aba Desempenho o Histórico de Uso da Memória Física; ou na aba Processos o valor associado ao seu programa na coluna Memória. Descreva o que ocorre.
IMPORTANTE: salve todos os seus trabalhos e não deixe nenhum conteúdo importante aberto em algum programa!!!

72. (*Cortesia da Profa. Leyza Baldo Dorini*) Um algoritmo genético é um procedimento computacional de busca inspirado no processo biológico de evolução que otimiza a solução de um problema. O problema é modelado por: uma população de indivíduos que representam possíveis soluções; uma função que avalia a qualidade da solução representada por cada indivíduo da população e um conjunto de operações genéticas. O procedimento consiste em aplicar os operadores genéticos sobre a população, gerando novos indivíduos, e selecionar os mais apropriados para constituírem a nova população. Esse processo é repetido até que uma solução adequada seja obtida. Dentre os operadores genéticos um importante é o de *crossover* de dois indivíduos. Esse operador corta em duas partes as sequências de genes de dois indivíduos pais (pai1 e pai2) e gera dois novos indivíduos filhos (filho1 e filho2), onde filho1 é dado pela concatenação da primeira parte dos genes de pai1 com a segunda parte dos genes de pai2, enquanto filho2 é dado pela concatenação da primeira parte de pai2 com a segunda parte de pai1. O diagrama abaixo exemplifica a operação de *crossover* em indivíduos representados por vetores de inteiros onde a primeira posição contém o tamanho do vetor.



Escreva uma função em C que execute a operação de *crossover* descrita acima apenas para o flho1. A função deve receber cinco parâametros: um vetor representando o primeiro pai, a posição de corte no primeiro pai, um vetor representando o segundo pai, a posição do corte no segundo pai, e um vetor que receberá o novo indivíduo. No exemplo apresentado a chamada do procedimento seria:

```
corte1 = 3; corte2 = 5;
crossover(pai1, corte1, pai2, corte2, filho1);
```

Note que os vetores devem iniciar na posição zero e essa posição é usada para armazenar o tamanho do vetor. No caso do exemplo, pai1[0]=11, pai2[0]=10, filho1[0]=8 e filho2[0]=13. O cabeçalho da função é:

```
void crossover(int *pai1, int corte1, int *pai2, int corte2, int*filho1);
```

73. Adaptar o Exercício 91 da lista (distâncias entre cidades) para:
    - pedir ao usuário a quantidade N de cidades na área desejada.
    - alocar dinamicamente uma matriz NxN.
    - pedir distâncias entre as cidades, conforme o enunciado original.
    - pedir ao usuário a quantidade M de etapas do percurso.
    - alocar dinamicamente um vetor de M elementos.
    - pedir o percurso conforme o enunciado original.
    - efetuar o cálculo de distância conforme o enunciado original.

Aula 7 – Arrays

Vetores

74. Faça um programa que dado o array 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 array

75. Escreva um programa para:
    a. pedir 10 números ao usuário e armazenar esses valores em um array
    b. pedir um outro número ao usuário e calcular e mostrar quantos números do array são inferiores a esse número.

    Fonte:http://eden.dei.uc.pt/~inf/Fichas/Exercicios_Java_2.pdf

76. Fazer um programa para:
    a. receber 10 números inteiros do usuário em um array
    b. copiar os pares para o array pares e os ímpares para o array ímpares, na ordem em que aparecem
    c. mostrar os arrays resultantes no final

    Ex: se o usuário forneceu

    7, 4, 2, 9, 3, 5, 0, 2, 3, 6

    o array pares deve conter: 4, 2, 0, 2, 6

    o array ímpares deve conter: 7, 9, 3, 5, 3

77. Faça um programa que:
    a. receba dez números de até 4 dígitos do usuário em um vetor adequado
    b. agrupe os números que são permutação uns dos outros, mostrando os grupos.

    Ex: para a digitação dos números:

    647 123 45 476 37 312 674 504 48 73

    O resultado deve ser:

    Grupo 1: 647, 476, 674
    Grupo 2: 123, 312
    Grupo 3: 45, 504
    Grupo 4: 37, 73
    Grupo 5: 48

78. Um determinado vetor de 60 posições contém os dados de velocidade instantânea de um veículo em m/s, obtidos ao longo de 60 segundos em intervalos de 1 segundo. Com base nestes dados, escreva um programa que mostre as seguintes informações:
    a. Qual o maior período de tempo em que o veículo se deslocou sem diminuir a velocidade.
    b. Qual o instante de tempo em que o veículo iniciou a frenagem mais abrupta.
    c. Qual a maior aceleração (positiva) à qual ele foi submetido e em qual instante de tempo ela se iniciou.
    d. Qual o maior período de tempo em que o veículo se deslocou com velocidade constante.

79. Ainda no contexto do exercício anterior, suponha que o usuário forneceu um vetor de 100 posições no qual cada posição corresponde à velocidade máxima em m/s na qual um veículo pode se deslocar em um determinado quarteirão (100 m). Com base neste vetor e no vetor fornecido no exercício anterior, calcular e mostrar:
    a. Em quantos quarteirões o veículo excedeu o limite de velocidade até 20%
    b. Em quantos quarteirões o veículo excedeu o limite de velocidade acima de 20%.
    c. O valor final devido em multas, considerando que existe um radar capaz de emitir uma multa a cada quarteirão, que a multa para excessos até 20% é de R$ 200,00, que a multa para excessos acima de 20% é de R$ 300,00, e que cada multa reincidente sofre um acréscimo de 50% em relação ao seu valor básico.

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

81. 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.

82. Pode-se calcular a área de um polígono a partir das coordenadas de seus vértices utilizando a seguinte fórmula:

    Area = soma (X(n)*Y(n+1) – X(n)*Y(n-1)) / 2

    sendo que:

    p/ n = 1, n-1 = N (número de vértices)

    p/ n = N, n+1 = 1

Escrever um programa em C que:

a. receba o número de vértices N do usuário
b. receba N pares de coordenadas (X,Y) dos vertíces do polígono em arrays adequados.
c. calcule e mostre a área segundo a fórmula acima.

83. 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.

84. (*Cortesia Prof. Bogdan Tomoyuki Nassu*) Um problema clássico da computação é o do embaralhamento: temos um número fixo de elementos distintos, e queremos colocá-los em uma ordem aleatória. Este problema é encontrado, por exemplo, em simulações de jogos de cartas. Para definir a ordem em que as cartas serão distribuídas, podemos usar um vetor com 52 posições, e colocar neste vetor, em uma ordem aleatória, os números 0 a 51. Cada número corresponde a uma carta do baralho.

    a. Suponha que você concebeu a seguinte solução para o problema: temos um vetor com 52 posições, inicialmente contendo "lixo". Até que tenhamos preenchido todas as posições com valores válidos, repetimos o seguinte procedimento: geramos um número aleatório entre 0 e 51, e inserimos o número gerado no vetor, mas apenas se este valor ainda não tiver sido gerado antes. No final, o vetor deve ter 52 valores diferentes entre si, em uma ordem aleatória. Como cada valor corresponde a uma carta do baralho, temos uma ordem aleatória para distribuir as cartas. Implemente este algoritmo.
    b. O problema do embaralhamento não aparece somente em jogos de cartas. Ele surge, por exemplo, em outras situações onde queremos simular cenários nos quais o estado inicial é desconhecido, e em alguns algoritmos probabilísticos da área de inteligência artificial. A solução apresentada no item a) tem um problema. Faça o seguinte teste: caso o seu programa ainda não faça isso, modifique-o para que ele imprima cada valor no momento em que ele for inserido no vetor. Modifique então o número de itens, de 52 para algum valor grande (mas não tão grande a ponto de o compilador apresentar um erro). Observe o tempo entre a impressão de dois números. O que acontece com este tempo? Por que?
    c. Tente conceber um algoritmo de embaralhamento mais eficiente. O objetivo é gerar um vetor que contenha N elementos distintos, com valores entre 0 e N-1, em uma ordem aleatória. Antes de procurar no Google ou consultar colegas, tente pensar a fundo no problema e chegar sozinho a uma solução - este é um daqueles exercícios nos quais pensar na solução é mais importante do que chegar nela!

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

86. (*Cortesia Prof. Bogdan Tomoyuki Nassu*) Em programas de transmissão de voz sobre IP (VoIP), como o Skype, o áudio é enviado em "pacotes". Os dados em um pacote sem compressão são representados como sinais de áudio: uma série de valores, com cada valor correspondendo a uma amostra do sinal em um instante do tempo

    Em conversas entre duas pessoas – incluindo conversas por telefone ou VoIP – é normal que um dos lados permaneça em silêncio enquanto o outro fala. Com base nisso, é possível aplicar uma ideia radical: para minimizar o uso da rede, pacotes contendo apenas silêncio (ou quase) simplesmente não são enviados!

    Crie uma função que recebe os seguintes parâmetros:

    `short* in`: buffer de entrada. Cada amostra é representada como um inteiro de 16 bits com sinal.

    `unsigned long length`: número de amostras no buffer de entrada.

    `unsigned short min_level`: amostras no intervalo `[-min_level, +min_level]` são consideradas como sendo silêncio. Esta tolerância é necessária porque o silêncio raramente é absoluto, devido a sons do ambiente ou mesmo ruído na captação.

    `double min_proportion`: valor entre 0 e 1, indica a proporção de amostras para que um sinal seja considerado como contendo silêncio. Normalmente, é um valor alto, acima de 0.90. Se a proporção de amostras indicar um número de amostras não-inteiro, o número de amostras deve ser arredondado para baixo. Por exemplo, se min_proportion é 0.98 e o buffer tem 240 amostras, um sinal com 235 ou mais amostras de silêncio será considerado como silêncio.

    A função deve retornar 1 se a proporção de amostras no intervalo `[-min_level, +min_level]` for maior ou igual a `min_proportion`, ou 0 do contrário. O protótipo da função é o seguinte:

    ```
    int ehSilencio (short* in, unsigned long length, unsigned short min_level,double min_proportion);
    ```

87. (*Cortesia Prof. Bogdan Tomoyuki Nassu*) Escreva um programa que recebe uma lista de datas digitadas pelo usuário e as mostra em ordem cronológica. Cada data é fornecida no formato DD/MM/AAAA. O programa deve ser capaz de receber até 64 datas, mas se o usuário digitar 0/0/0, o programa deve imediatamente mostrar as datas digitadas ordenadas, sem considerar o 0/0/0. Neste exercício, não se preocupe em verificar a consistência / validade das entradas.

Matrizes

88. Faça um programa que, dados dois arrays 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

89. 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.

90. 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.

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

	A	B	C	D	E
A	00	15	30	05	12
B	15	00	10	17	28
C	30	10	00	03	11
D	05	17	03	00	80
E	12	28	11	80	00

Implemente um programa que:

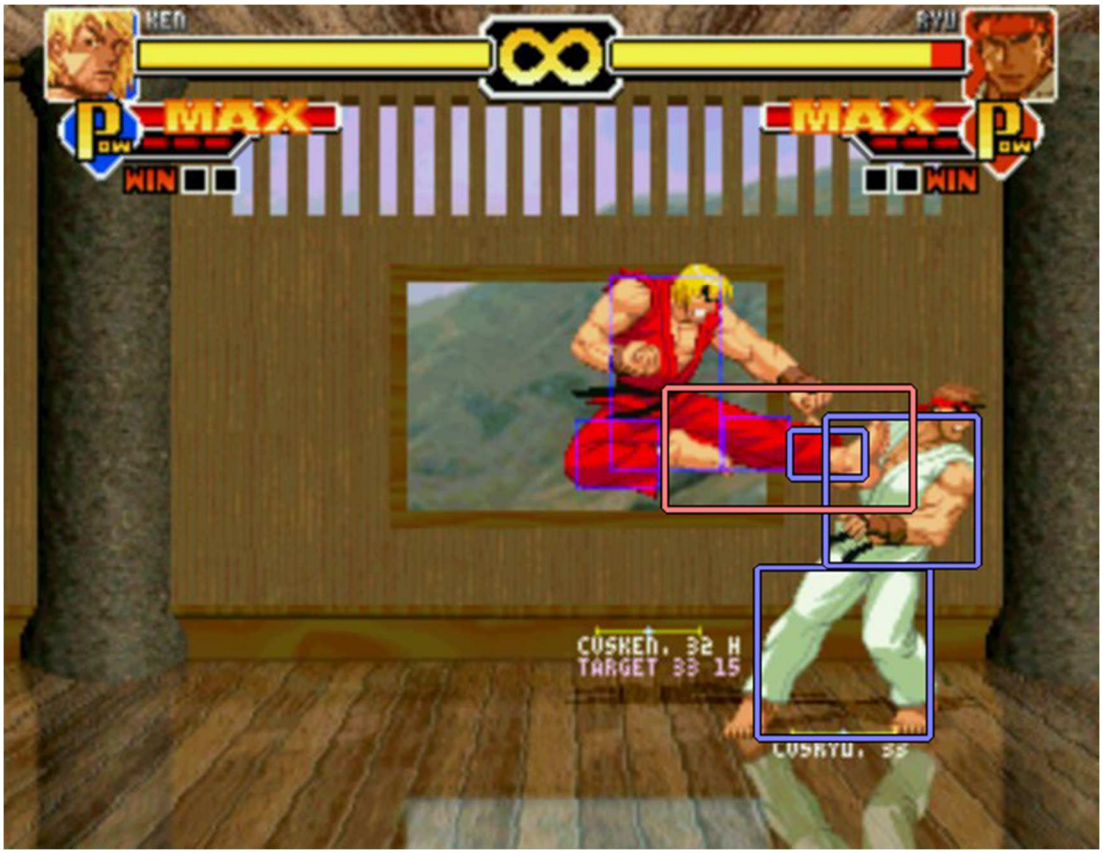
a. obtenha do usuário uma tabela semelhante à acima e armazene-a em um *array* bidimensional (matriz). O programa não deve perguntar distâncias já informadas (por exemplo, se o usuário já forneceu a distância entre A e C não é necessário informar a distância entre C e A, que é a mesma) e também não deve perguntar a distância de uma cidade para ela mesma, que é 0.
b. obtenha um percurso fornecido pelo usuário (máx. 10 cidades visitadas) em um *array* unidimensional (vetor).
c. Calcule e mostre a distância percorrida. Por exemplo: dado o percurso A, B, C, B, E, A, D, para a tabela mostrada como exemplo teremos: 15 + 10 + 10 + 28 + 12 + 5 = 80 km.

92. 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).

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

94. (*Cortesia Prof. Bogdan Tomoyuki Nassu*) Um problema encontrado na programação de jogos de computador é a detecção de colisões entre objetos. Uma solução para este problema se baseia em "caixas de colisão". A ideia é representar cada objeto como um conjunto de retângulos. Colisões são detectadas quando há sobreposições entre retângulos pertencentes a objetos distintos.

    Por exemplo, na figura a seguir o "golpe" é representado por um retângulo, e o "personagem golpeado" é representado por 3 retângulos. Podemos detectar a colisão porque o retângulo que representa o golpe se sobrepõe a 2 dos retângulos que representam o personagem golpeado. Cada retângulo é definido por 2 pontos (x1, y1) e (x2, y2) em um plano cartesiano, onde (x1, y1) é o canto superior esquerdo e (x2, y2) é o canto inferior direito. Na figura do exemplo, o retângulo que representa o golpe é definido pelas coordenadas (388, 260) e (532, 194), supondo que a origem do plano está no canto inferior esquerdo da imagem.



a. Escreva uma função que receba os seguintes parâmetros:

`int golpe [4]`: Retângulo que representa o golpe. Os 4 valores dados se referem a x1, y1, x2 e y2, nesta ordem.

`int* adversario`: Um vetor contendo os retângulos que representam o adversário do personagem que desferiu o golpe. Cada retângulo é dado como uma sequência de 4 valores, referentes a x1, y1, x2 e y2, nesta ordem.

`int n_retangulos`: O número de retângulos que representam o adversário do personagem que desferiu o golpe.

A função deve retornar 1 se o golpe atingiu o adversário, ou 0 do contrário. Você pode supor que os dados passados para a função não contêm erros. O protótipo da função é o seguinte:

```
int atingiu (int golpe [4], int* adversario, int n_retangulos);
```

Você pode invocar a função *sobrepoe()* especificada na letra b), mesmo que não a tenha implementado – simplesmente suponha que ela existe e funciona.

b. Implemente uma função que recebe 2 retângulos e retorna 1 caso exista uma sobreposição entre eles, ou 0 do contrário. Cada retângulo é representado como um conjunto de 4 valores: x1, y1, x2 e y2, sendo que o ponto (x1, y1) é o canto superior esquerdo do retângulo, e (x2, y2) é o seu canto inferior direito. O protótipo da função é o seguinte:

```
int sobrepoe (int retangulo1 [4], int retangulo2 [4]);
```

95. (*Cortesia Prof. Bogdan Tomoyuki Nassu*) O Sudoku é um quebra-cabeça numérico que tem como objetivo preencher com valores de 1 a 9 um conjunto de 9x9 células em um grid, com base em alguns números iniciais e algumas regras: não podem existir números repetidos em uma mesma linha, nem em uma mesma coluna. Além disso, as células são divididas em 9 subconjuntos de 3x3 células, e não podem existir números repetidos em um mesmo subconjunto. A figura abaixo mostra um quebra-cabeça resolvido:

9	5	4	8	1	6	3	7	2
7	8	6	2	5	3	1	4	9
1	2	3	7	9	4	6	5	8
3	1	8	9	7	2	4	6	5
2	7	9	4	6	5	8	1	3
4	6	5	3	8	1	9	2	7
8	4	7	1	2	9	5	3	6
5	3	2	6	4	8	7	9	1
6	9	1	5	3	7	2	8	4

Escreva um programa que recebe uma solução de um quebra-cabeça como uma matriz de 9x9 inteiros. O programa deve dizer se a matriz contém uma solução válida de acordo com as regras do Sudoku. Para evitar o trabalho de ter que digitar os números a cada execução do programa, inicialize a matriz no próprio código, como no exemplo abaixo:

```
int resposta_sudoku [9][9] =
        {{9, 5, 4, 8, 1, 6, 3, 7, 2},
         {7, 8, 6, 2, 5, 3, 1, 4, 9},
         {1, 2, 3, 7, 9, 4, 6, 5, 8},
         {3, 1, 8, 9, 7, 2, 4, 6, 5},
         {2, 7, 9, 4, 6, 5, 8, 1, 3},
         {4, 6, 5, 3, 8, 1, 9, 2, 7},
         {8, 4, 7, 1, 2, 9, 5, 3, 6},
         {5, 3, 2, 6, 4, 8, 7, 9, 1},
         {6, 9, 1, 5, 3, 7, 2, 8, 4}};
```

96. (*Cortesia Prof. Bogdan Tomoyuki Nassu)* Os incas construíam pirâmides de base quadrada nas quais a única forma de se atingir o topo era seguir em espiral pela borda, que acabava formando uma rampa em espiral*. Uma “matriz inca” é uma matriz quadrada NxN de números inteiros na qual, partindo do canto superior esquerdo, no sentido horário, em espiral, a posição seguinte na ordem é o inteiro consecutivo da posição anterior. Por

exemplo, as matrizes abaixo são incas:

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

Escreva um programa que gera uma matriz quadrada e a preenche de forma a gerar uma "matriz inca". O número de linhas/colunas deve ser fornecido pelo usuário. A geração da matriz deve ser feita em uma função com o seguinte protótipo:

```
void geraMatrizInca (int** matriz, int lado);
```

*Esta informação não foi verificada, mas não afeta o exercício.

97. (*Cortesia Prof. Bogdan Tomoyuki Nassu)* Um quadrado mágico é uma matriz quadrada na qual a soma dos elementos de cada linha, a soma dos elementos de cada coluna, e a soma dos elementos das diagonais são todos iguais. Por exemplo, a matriz abaixo é um quadrado mágico, pois 8+0+7 = 4+5+6 = 3+10+2 = 8+4+3 = 0+5+10 = 7+6+2 = 8+5+2 = 3+5+7 = 15:

    $$\begin{bmatrix} 8 & 0 & 7 \\ 4 & 5 & 6 \\ 3 & 10 & 2 \end{bmatrix}$$

    Escreva uma função que recebe uma matriz quadrada, alocada dinamicamente, e retorna 1 se ela é um quadrado mágico, ou 0 do contrário.

98. Fazer um programa que:
    a. recebe do usuário uma matriz 20x20. Esta matriz corresponde ao mapa de um labirinto quadrado, sendo que o número 1 indica uma posição onde existe parede e o número 0 indica uma posição onde existe um corredor.
    b. calcula e informa se todas as posições de corredor são alcançáveis por um personagem posicionado inicialmente em um local qualquer.

Aula 8 – Strings

99. Faça um programa que receba um nome completo do usuário e retorne a abreviatura deste nome. Não se devem abreviar palavras com dois caracteres ou menos, tais como as preposições: do, de, etc. A abreviatura deve vir separada por pontos. Ex: Paulo Jose de Almeida Prado. Abreviatura: P. J. de A. P.

100. Fazer um programa em C para:
    a. receber um nome do usuário na forma de string (máx. 100 caracteres).
    b. mostrar o nome no formato de autor de referência bibliográfica (último sobrenome no início, depois as iniciais dos demais nomes, eliminando todas as palavras com dois caracteres ou menos).

Por exemplo, para o nome "Paulo Jose de Almeida Prado", o resultado seria "Prado, P. J. A."

101. 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, implementando uma função apropriada para esta finalidade.

102. 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**

103. 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

104. 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.

105. 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.

106. 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".

107. Fazer um programa que efetua teste de velocidade de digitação:
a. Programa mostra ao usuário uma palavra aleatória, obtida de uma lista pré-definida.
b. Usuário tem que digitar aquela palavra e pressionar ENTER ou espaço.
c. ao final de 1 minuto, o jogo deve mostrar quantas palavras corretas e quantas palavras erradas o usuário digitou.

108. 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.

109. 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.

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

111. 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.

112. 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.

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

114. 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.

115. 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 K e retorne a mensagem criptografada pelo código de César. Fraquezas: apenas 26 chaves possíveis. É possível utilizar conhecimento da linguagem para facilitar a busca.

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

117. Faça um programa que inverta a ordem das letras de uma string. Utilize o código do exercício 115 para incrementar o código de César adicionando a inversão da mensagem encriptada.

118. 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.

119. 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.

Em seguida, escrever um programa que receba duas strings do usuário, o valor da posição, chame a função anteriormente implementada e exiba o resultado ao usuário.

120. 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

121. *(Cortesia do Prof. Bogdan Tomoyuki Nassu)* Escreva uma função que recebe uma string e retorna 1 se ela contém um palíndromo, ou 0 do contrário. Um palíndromo é uma sequência que pode ser lida igualmente da esquerda para a direita ou da direita para esquerda. Exemplos de palíndromos: "ovo", "esse", "eva asse essa ave".

122. *(Cortesia do Prof. Bogdan Tomoyuki Nassu)* Cebolinha é um personagem de história em quadrinhos que quando fala, troca o R pelo L. Escreva uma função *cebolinhaString*, que recebe uma string s e um buffer (um vetor capaz de armazenar outra string com o mesmo tamanho de s) e armazene no buffer uma versão de s com todos os R's trocados por L's, exceto quando o R é a última letra de uma palavra. Dois R's seguidos devem ser substituídos por um único L. Lembre-se de manter as letras maiúsculas e minúsculas como no texto original.

Aula 9 – Registros

123. 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.

124. Fazer um programa de diálogo de login semelhante ao exercício 106, 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.

125. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Escreva um programa que usa struct's para armazenar dados de um estacionamento. Use uma struct para armazenar a placa e o modelo de um carro, assim como o horário de entrada e saída do estacionamento, com horas e minutos. Defina e preencha uma variável do tipo da struct (= 1 carro), e mostre os dados do carro junto com o valor devido ao estacionamento (preços: R$5,00 a primeira hora, R$2,00 para cada hora extra, valores proporcionais ao tempo para horas incompletas).

126. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Suponha que você está fazendo uma pesquisa de preços para a compra de um tablet. Para auxiliá-lo, escreva um programa que permite o cadastro de vários registros, cada um contendo o nome de uma loja, o seu telefone, e o preço do tablet naquela loja. Os dados para cada loja podem ser fornecidos no próprio código. O programa deve calcular a média dos preços encontrados e mostrar uma relação contendo o nome e o telefone das lojas que tinham o tablet com preço abaixo da média.

127. (*Cortesia do Prof. Bogdan Tomoyuki Nassu*) Escreva uma função que recebe 2 parâmetros do tipo Horario (com campos para hora, minutos e segundos), h1 e h2, e retorna o número de segundos passados entre h1 e h2, supondo que h2 ocorre depois de h1.

```
int segundosEntre (Horario h1, Horario h2);
```

128. (*Cortesia do Prof. Bogdan Tomoyuki Nassu)* Escreva uma função que recebe 3 float's e retorna uma struct que tem 3 campos do tipo float: menor, maior e meio. A função deve analisar os valores e preencher a struct.

```
MaMeMe classifica (float n1, float n2, float n3);
```

129. (*Cortesia do Prof. Bogdan Tomoyuki Nassu)* Escreva uma versão da função anterior que ordena os valores já colocados em uma variável passada por referência. Os valores são dados em uma ordem arbitrária, e reordenados pela função.

```
void classifica (MaMeMe* valores);
```

130. (*Cortesia da Profa. Leyza Baldo Dorini*) Dados os seguintes campos de um registro: nome, dia de aniversário e mês de aniversário, desenvolver um programa que mostre para

cada um dos meses do ano quem são as pessoas que fazem aniversário e também a idade atual de cada pessoa (pedir ao usuário a data de hoje). Considere um conjunto de 40 pessoas.

131. Fazer um programa para gerenciamento de compras em um supermercado, com as seguintes características:

a. O programa deve mostrar um menu de opções ao usuário: adicionar item, remover item, obter valor total da compra, mostrar relatório (lista de itens) e sair.
b. O programa deve definir o tipo de registro *Item*, que pretende representar um item sendo comprado no supermercado.
    i. Campos: nome do item, valor unitário e quantidade.
c. O programa deve definir um array (vetor) para permitir criar uma lista de compras no um supermercado. O tamanho inicial do array deve ser 5 e deve aumentar à medida que mais do que 5 elementos sejam adicionados à lista (dica: utilizar alocação e desalocação dinâmicas)

Aula 10 – Arquivos

132. Fazer um programa em C que:
a. Receba o nome, o código e as duas notas bimestrais de N alunos para uma determinada matéria.
b. Salve estes dados em um arquivo. Os dados devem ser salvos registro a registro, obedecendo o seguinte formato:
    i. número inteiro contendo o tamanho em char do nome do aluno.
    ii. sequência de chars correspondente à string que contém o nome do aluno.
    iii. código na forma de número inteiro
    iv. duas notas na forma de números inteiros.

Fazer um programa em C para:
c. ler os dados contidos no arquivo gerado pelo programa anterior
d. calcular e mostrar: quais alunos foram aprovados, quais foram reprovados e a média da turma.

133. Uma loja possui 4 filiais, cada uma com um código de 1 a 4. Um arquivo contendo todas as vendas feitas durante o dia nas quatro filiais é gerado a partir de uma planilha, sendo que cada linha do arquivo contém o número da filial e o valor de uma venda efetuada, separados por vírgula. Ex.:

1,189.90
1,45.70
3,29.90
4,55.00

No exemplo acima, a filial 1 fez duas vendas, a primeira totalizando R$ 189,90 e a segunda R$ 45,70. A filial 3 fez uma venda de R$ 29,90 e a 4 também uma de R$ 55,00. Faça um programa que leia este arquivo e informe, ao final, o total e o valor médio das vendas de cada filial.
Fonte:http://74.125.47.132/search?q=cache:uMJJLyJh1CwJ:www.disi.unitn.it/~vitorsouza/sites/default/files/JavaSwingBDExerciciosRevisao.pdf+exerc%C3%ADcios+java&cd=20&hl=pt-BR&ct=clnk&client=opera

134. Implementar um programa de gerenciamento de high scores. O programa deve:
   a. carregar os high scores de um arquivo (de texto ou binário) 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 de texto ou binário.
   d. Mostrar a tabela de highscores atualizada.

   Obs: a tabela de highscores pode ser implementada na forma de um array (vetor) de registros que contêm os dados de um score (nome e pontuação).

135. Reimplementar o programa do exercício 131 para que este carregue a última lista de compras de um arquivo, ao iniciar, e salve a lista atualizada no arquivo quando a opção "Sair" for selecionada.

136. A CEF disponibiliza um arquivo online com todos os resultados da mega-sena, organizados na forma de uma tabela. Considerando que temos este arquivo na forma de texto, fazer um programa que efetue as seguintes operações:
   a. Receba um jogo do usuário (6 números, de 1 a 60) e verifique se aquele jogo já foi sorteado e quando foi sorteado.
   b. Calcule e mostre quais os 10 números mais sorteados de todos os tempos
   c. Mostre qual o maior prêmio e o menor prêmio totais já rateados

137. Adaptar o exercício 133 da lista (vendas) para:
   a. definir um programa que escreve o registro de vendas na forma binária. Este programa pergunta ao usuário o código da filial, o valor da venda e grava no arquivo, enquanto o código da filial for válido (1 a 4).
   b. definir um programa que lê os dados na forma binária e faz o mesmo processamento do enunciado original.