

Exercícios: Revisão de C

# *Tópico 1 – Aspectos básicos*

1. Fazer um programa para
   1. receber do usuário um tempo em segundos, correspondente à duração de um evento qualquer (por ex. jogo de futebol)
   2. 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:
   1. 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
   2. Calcular e mostrar a quantos metros de distância o projétil atingirá o chão, considerando aceleração da gravidade igual a 9,81 m/s2.

# *Tópico 2 – Estruturas condicionais*

1. Fazer um programa para ler um número do usuário e determinar se este número é par ou não par.
2. 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.
3. Fazer um programa que recebe 3 valores não inteiros do usuário e mostra o maior deles, o menor deles e o valor intermediário.
4. 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.
5. 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).
6. 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>
7. (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.
8. 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.
9. (*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.

1. 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.
2. (*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.

1. (*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.

1. 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:
   1. Triângulo: A = L\*L\*1.73/4
   2. Quadrado: A = L\*L
   3. Hexágono: A = 6\*L\*L\*1.73/4;

# *Tópico 3 - Estruturas de repetição*

1. 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.
2. Fazer um programa para mostrar os 15 primeiros termos da série de Fibonacci.
3. 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.
4. Fazer um programa para receber um número inteiro do usuário e determinar se este número é primo ou não.
5. 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).
6. (*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.
7. Fazer um programa para receber um número do usuário e decompô-lo em fatores primos.
8. 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.
9. Fazer um programa para receber dois números inteiros do usuário e mostrar o seu MMC (mínimo múltiplo comum).
10. 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).
11. Fazer um programa para medir os reflexos do usuário. O programa deve:
    1. Mostrar a palavra “Agora!” após um tempo aleatório e um número, também aleatório
    2. 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),

1. Fazer um programa para mostrar a soma de todos os números 4 do dominó.
2. 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.
3. (*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.
4. 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.*
5. 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.
6. 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.
7. 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:
   1. De 1 a 3 tentativas: muito sortudo
   2. De 4 a 6 tentativas: sortudo
   3. De 7 a 10 tentativas: normal
   4. > 10 tentativas: tente novamente
8. Escrever um programa para ler um número inteiro do usuário e exibir o maior número primo que seja menor do que o número digitado.
9. Fazer um programa para exibir os n primeiros múltiplos simultâneos de dois números dados.
10. (*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.
11. 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.
12. 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.
    1. 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).
    2. 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*?
    3. 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.
13. 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.*
14. (*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:
    1. A cada ano ele planta todos os grãos da colheita anterior.
    2. Há apenas uma colheita por ano
    3. 0% (dez por cento) dos grãos não germinam (morrem sem produzir).
    4. Cada grão que germina produz duas espigas de milho.

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

1. 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:
   1. 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).
   2. 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.

# *Tópico 4 – Funções*

1. Fazer uma função *situacao\_aluno* que:
   1. recebe como parâmetros: média final do aluno (0 a 100), número de faltas e quantidade de horas-aula no semestre
   2. 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.

1. Implementar a função *RaizQuadrada*. Esta função deve:
   1. Receber um número do tipo *float* como parâmetro.
   2. 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.
   3. Não utilize outras funções prontas (sqrt, pow) – deve implementar utilizando um método numérico.
2. Implementar a função *inverte* que recebe um número *unsigned int* como parâmetro e retorna este número escrito ao contrário. Ex: 431 <-> 134.
3. Implementar a função *double power (double base, double expoente)*, que retorna o valor de

*base* elevado a *expoente.* Dicas:

* 1. Transformar o expoente em uma razão de inteiros (multiplicando ambos por 10 até o numerador ficar inteiro).
  2. Simplificar a razão de inteiros com sucessivas divisões de numerador e denominador.
  3. Calcular *basenumerador*
  4. Calcular (*basenumerador*)*1/denominador* . Utilizar a função de raiz anteriormente implementada.
  5. Se *basenumerador* estourar a faixa dos *double*, dividir *numerador* e *denominador* por 10 e repetir.
  6. Se *numerador* for negativo, resultado é 1/resultado.

1. 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.
2. 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.
3. (*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);

1. (*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).

1. (*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”).

# *Tópico 5 – Recursividade*

Bibliografia auxiliar: https://programacaodescomplicada.files.wordpress.com/2012/10/lista- recursc3a3o.pdf

1. Escreva a função para cálculo do N-ésimo termo da série de Fibonacci utilizando recursividade.
2. Implementar a função EXP com as seguintes características:
   1. Recebe um valor de *base* e um valor de *expoente* como parâmetros do tipo *unsigned int*.
   2. Retorna o resultado de *baseexpoente* na forma de um *unsigned int.*
   3. Armazena o valor 1 na variável global *status* se foi bem sucedida e 0 se a exponenciação não pôde ser calculada devido a estouro.
   4. Utiliza a característica recursiva da exponenciação:
      1. *Baseexp* = *base.baseexp-1* , *exp > 0*
      2. *Baseexp = 1, exp ==0*
3. Considere uma partida de futebol entre duas equipes A x B, cujo placar final é m x n, em que *m* e *n* são os 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”.
4. 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 sequência de movimentos para resolver o problema da Torre de Hanói.
5. 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.
6. Crie uma função recursiva que receba um número inteiro positivo N e calcule o somatório dos números de 1 a N.
7. 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.
8. Dado um número *n* na base decimal, escreva uma função recursiva em C que imprime este número na base binária.

# *Tópico 6 - Vetores*

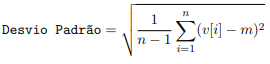
* 1. Faça um programa que possua um vetor denominado *A* que armazene 6 números inteiros. O programa deve executar os seguintes passos:
     1. Atribua os seguintes valores a esse vetor: 1, 0, 5, -2, -5, 7.
     2. Armazene em uma variável inteira (simples) a soma entre os valores das posições

*A*[0], *A*[1] e *A*[5] do vetor e mostre na tela esta soma.

* + 1. Modifique o vetor na posição 4, atribuindo a esta posição valor 100.
    2. Mostre na tela cada valor do vetor *A*, um em cada linha.
  1. Crie um programa que lê 6 valores inteiros e, em seguida, mostre na tela os valores lidos.
  2. Ler um conjunto de números reais, armazenando-o em vetor e calcular o quadrado das componentes deste vetor, armazenando o resultado em outro vetor. Os conjuntos têm 10 elementos cada. Imprimir todos os conjuntos.
  3. Faça um programa que leia um vetor de 8 posições e, em seguida, leia também dois valores *X* e *Y* quaisquer correspondentes a duas posições no vetor. Ao final, seu programa deverá escrever a soma dos valores encontrados nas respectivas posições *X* e *Y* .
  4. Leia um vetor de 10 posições. Contar e escrever quantos valores pares ele possui.
  5. Faça um programa que receba do usuário um vetor com 10 posições. Em seguida, deverá ser impresso o maior e o menor elemento do vetor.
  6. Escreva um programa que leia 10 números inteiros e os armazene em um vetor. Imprima o vetor, o maior elemento e a posição que ele se encontra.
  7. Crie um programa que lê 6 valores inteiros e, em seguida, mostre na tela os valores lidos na ordem inversa.
  8. Crie um programa que lê 6 valores inteiros pares e, em seguida, mostre na tela os valores lidos na ordem inversa.
  9. Faça um programa para ler a nota da prova de 15 alunos e armazene num vetor, calcule e imprima a média geral.
  10. Faça um programa que preencha um vetor com 10 números reais, calcule e mostre a quantidade de números negativos e a soma dos números positivos desse vetor.
  11. Fazer um programa para ler 5 valores e, em seguida, mostrar todos os valores lidos juntamente com o maior, o menor e a média dos valores.
  12. Fazer um programa para ler 5 valores e, em seguida, mostrar a posição onde se encontram o maior e o menor valor.
  13. Faça um programa que leia um vetor de 10 posições e verifique se existem valores iguais e os escreva na tela.
  14. Leia um vetor com 20 números inteiros. Escreva os elementos do vetor eliminando elementos repetidos.
  15. Faça um programa que leia um vetor de 5 posições para números reais e, depois, um código inteiro. Se o código for zero, finalize o programa; se for 1, mostre o vetor na ordem direta; se for 2, mostre o vetor na ordem inversa. Caso o código for diferente de 1 e 2 escreva uma mensagem informando que o código é invalido.
  16. Leia um vetor de 10 posições e atribua valor 0 para todos os elementos que possuírem valores negativos.
  17. Faça um programa que leia um vetor de 10 números. Leia um número *x*. Conte os múltiplos de um número inteiro *x* num vetor e mostre-os na tela.
  18. Faça um vetor de tamanho 50 preenchido com o seguinte valor: (*i* + 5 \* *i*)%(*i* + 1), sendo

i, a posição do elemento no vetor. Em seguida imprima o vetor na tela.

* 1. Escreva um programa que leia números inteiros no intervalo [0,50] e os armazene em um vetor com 10 posições. Preencha um segundo vetor apenas com os números ímpares do primeiro vetor. Imprima os dois vetores, 2 elementos por linha.
  2. Faça um programa que receba do usuário dois vetores, *A* e *B*, com 10 números inteiros cada. Crie um novo vetor denominado *C* calculando *C* = *A* - *B*. Mostre na tela os dados do vetor *C*.
  3. Faça um programa que leia dois vetores de 10 posições e calcule outro vetor contendo, nas posições pares os valores do primeiro e nas posições ímpares os valores do segundo.
  4. Ler dois conjuntos de números reais, armazená-los em vetores e calcular o produto escalar entre eles. Os conjuntos têm 5 elementos cada. Imprimir os dois conjuntos e o produto escalar, sendo que o produto escalar é dado por: *x*1 ∗ *y*1 + *x*2 ∗ *y*2 + *...* + *xn* ∗ *yn*.
  5. Faça um programa que leia dez conjuntos de dois valores, o primeiro representando o número do aluno e o segundo representando a sua altura em metros. Encontre o aluno mais baixo e o mais alto. Mostre o número do aluno mais baixo e do mais alto, juntamente com suas alturas.
  6. Faça um programa que preencha um vetor de tamanho 100 com os 100 primeiros naturais que não são múltiplos de 7 ou que terminam com 7.
  7. Faça um programa que calcule o desvio padrão de um vetor *v,* contendo *n* = 10 números, onde *m* é a média do vetor.



* 1. Leia 10 números inteiros e armazene em um vetor. Em seguida escreva os elementos que são primos e suas respectivas posições no vetor.
  2. Leia 10 números inteiros e armazene em um vetor *v*. Crie dois novos vetores *v*1 e *v*2. Copie os valores ímpares de *v* para *v*1, e os valores pares de *v* para *v*2. Note que cada um dos vetores *v*1 e *v*2 teˆm no máximo 10 elementos, mas nem todos os elementos são utilizados. No final escreva os elementos UTILIZADOS de *v*1 e *v*2.
  3. Faça um programa que receba 6 números inteiros e mostre:
* Os números pares digitados;
* A soma dos números pares digitados;
* Os números ímpares digitados;
* A quantidade de números ímpares digitados;
  1. Faça um programa que leia dois vetores de 10 elementos. Crie um vetor que seja a intersecção entre os 2 vetores anteriores, ou seja, que contém apenas os números que estão em ambos os vetores. Não deve conter números repetidos.
  2. Faça um programa que leia dois vetores de 10 elementos. Crie um vetor que seja a união entre os 2 vetores anteriores, ou seja, que contém os números dos dois vetores. Não deve conter nu´meros repetidos.
  3. Leia dois vetores de inteiros *x* e *y*, cada um com 5 elementos (assuma que o usuário não informa elementos repetidos). Calcule e mostre os vetores resultantes em cada caso abaixo:
* Soma entre *x* e *y*: soma de cada elemento de *x* com o elemento da mesma posição em *y*.
* Produto entre *x* e *y*: multiplicação de cada elemento de *x* com o elemento da mesma posição em *y*.
* Diferença entre *x* e *y*: todos os elementos de *x* que não existam em *y*.
* Interseção entre *x* e *y*: apenas os elementos que aparecem nos dois vetores.
* União entre *x* e *y*: todos os elementos de *x*, e todos os elementos de *y* que não estão em *x*.
  1. Faça um programa que leia um vetor de 15 posições e o compacte, ou seja, elimine as posições com valor zero. Para isso, todos os elementos à frente do valor zero, devem ser movidos uma posição para trás no vetor.
  2. Faça um programa para ler 10 nu´meros DIFERENTES a serem armazenados em um vetor. Os dados deverão ser armazenados no vetor na ordem que forem sendo lidos, sendo que caso o usuário digite um nu´mero que já foi digitado anteriormente, o programa deverá pedir para ele digitar outro nu´mero. Note que cada valor digitado pelo usuário deve ser pesquisado no vetor, verificando se ele existe entre os números que já foram fornecidos. Exibir na tela o vetor final que foi digitado.
  3. Faça um programa que leia dois números *a* e *b* (positivos menores que 10000) e:
* Crie um vetor onde cada posição é um algarismo do número. A primeira posição é o algarismo menos significativo;
* Crie um vetor que seja a soma de *a* e *b*, mas faça-o usando apenas os vetores construídos anteriormente.

Dica: some as posições correspondentes. Se a soma ultrapassar 10, subtraia 10 do resultado e some 1 a próxima posição.

* 1. Leia um vetor com 10 números reais, ordene os elementos deste vetor, e no final escreva os elementos do vetor ordenado.
  2. Considere um vetor *A* com 11 elementos onde *A*1 *< A*2 *< … < A*6 *> A*7 *> A*8 *> … > A*11, ou seja, está ordenado em ordem crescente até o sexto elemento, e a partir desse elemento está ordenado em ordem decrescente. Dado o vetor da questão anterior, proponha um algoritmo para ordenar os elementos.
  3. Peça ao usuário para digitar dez valores numéricos e ordene por ordem crescente esses valores, guardando-os num vetor. Ordene o valor assim que ele for digitado. Mostre ao final na tela os valores em ordem.
  4. Escreva um programa que leia um número inteiro positivo *n* e em seguida imprima *n*

linhas do chamado Triângulo de Pascal:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

...

# *Tópico 7 - Matrizes*

* 1. Leia uma matriz 4 x 4, conte e escreva quantos valores maiores que 10 ela possui.
  2. Declare uma matriz 5 x 5. Preencha com 1 a diagonal principal e com 0 os demais elementos. Escreva ao final a matriz obtida.
  3. Fac¸a um programa que preenche uma matriz 4 x 4 com o produto do valor da linha e da coluna de cada elemento. Em seguida, imprima na tela a matriz.
  4. Leia uma matriz 4 x 4, imprima a matriz e retorne a localização (linha e a coluna) do maior valor.
  5. Leia uma matriz 5 x 5. Leia também um valor *X*. O programa deverá fazer uma busca desse valor na matriz e, ao final, escrever a localização (linha e coluna) ou uma mensagem de "não encontrado”.
  6. Leia duas matrizes 4 x 4 e escreva uma terceira com os maiores valores de cada posição das matrizes lidas.
  7. Gerar e imprimir uma matriz de tamanho 10 x 10, onde seus elementos são da forma:
* *A*[*i*][*j*] = 2*i* + 7*j* − 2 se *i < j*;
* *A*[*i*][*j*] = 3*i*2 − 1 se *i* = *j*;
* *A*[*i*][*j*] = 4*i*3 − 5*j*2 + 1 se *i > j.*
  1. Leia uma matriz de 3 x 3 elementos. Calcule a soma dos elementos que estão acima da diagonal principal.
  2. Leia uma matriz de 3 x 3 elementos. Calcule a soma dos elementos que estão abaixo da diagonal principal.
  3. Leia uma matriz de 3 x 3 elementos. Calcule a soma dos elementos que estão na diagonal principal.
  4. Leia uma matriz de 3 x 3 elementos. Calcule a soma dos elementos que estão na diagonal secundária.
  5. Leia uma matriz de 3 x 3 elementos. Calcule e imprima a sua transposta.
  6. Gere matriz 4 x 4 com valores no intervalo [1, 20]. Escreva um programa que transforme a matriz gerada numa matriz triangular inferior, ou seja, atribuindo zero a todos os elementos acima da diagonal principal. Imprima a matriz original e a matriz transformada.
  7. Faça um programa para gerar automaticamente nu´meros entre 0 e 99 de uma cartela de bingo. Sabendo que cada cartela deverá conter 5 linhas de 5 números, gere estes dados de modo a não ter números repetidos dentro das cartelas. O programa deve exibir na tela a cartela gerada.
  8. Leia uma matriz 5 x 10 que se refere às respostas de 10 questões de múltipla escolha, referentes a 5 alunos. Leia também um vetor de 10 posições contendo o gabarito de respostas que podem ser *a*, *b*, *c* ou *d*. Seu programa deverá comparar as respostas de cada candidato com o gabarito e emitir um vetor denominado resultado, contendo a pontuação correspondente a cada aluno.
  9. Faça um programa para corrigir uma prova com 10 questões de múltipla escolha (*a*, *b*, *c*, *d* ou *e*), em uma turma com 3 alunos. Cada questão vale 1 ponto. Leia o gabarito, e para cada aluno leia sua matrícula (número inteiro) e suas respostas. Calcule e escreva: Para cada aluno, escreva sua matrícula, suas respostas, e sua nota. O percentual de aprovação, assumindo média 7.0.
  10. Leia uma matriz 10 x 3 com as notas de 10 alunos em 3 provas. Em seguida, escreva o número de alunos cuja pior nota foi na prova 1, o número de alunos cuja pior nota foi na prova 2, e o número de alunos cuja pior nota foi na prova 3. Em caso de empate das piores notas de um aluno, o critério de desempate é arbitrário, mas o aluno deve ser contabilizado apenas uma vez.
  11. Faça um programa que permita ao usuário entrar com uma matriz de 3 x 3 números inteiros. Em seguida, gere um *array* unidimensional pela soma dos números de cada coluna da matriz e mostrar na tela esse *array*. Por exemplo, a matriz:

| 5 | -8 | 10 |
| --- | --- | --- |
| 1 | 2 | 15 |
| 25 | 10 | 7 |

Vai gerar um vetor, onde cada posição é a soma das colunas da matriz. A primeira posição será 5 + 1 + 25, e assim por diante:

31 4 3

* 1. Faça um programa que leia uma matriz de 5 linhas e 4 colunas contendo as seguintes informações sobre alunos de uma disciplina, sendo todas as informações do tipo inteiro:
* **Primeira coluna:** número de matrícula (use um inteiro)
* **Segunda coluna:** média das provas
* **Terceira coluna:** média dos trabalhos
* **Quarta coluna:** nota final

Elabore um programa que:

Leia as três primeiras informações de cada aluno

* + 1. Calcule a nota final como sendo a soma da média das provas e da média dos trabalhos
    2. Imprima a matrícula do aluno que obteve a maior nota final (assuma que só existe uma maior nota)
    3. Imprima a média aritmética das notas finais
  1. Faça um programa que leia uma matriz 3 x 6 com valores reais.
     1. Imprima a soma de todos os elementos das colunas ímpares.
     2. Imprima a média aritmética dos elementos da segunda e quarta colunas.
     3. Substitua os valores da sexta coluna pela soma dos valores das colunas 1 e 2.
     4. Imprima a matriz modificada.
  2. Faça um programa que leia duas matrizes 2 x 2 com valores reais. Ofereça ao usuário um menu de opções:
     1. somar as duas matrizes
     2. subtrair a primeira matriz da segunda
     3. adicionar uma constante às duas matrizes
     4. imprimir as matrizes

Nas duas primeiras opções uma terceira matriz 3 x 3 deve ser criada. Na terceira opção valor da constante deve ser lido e o resultado da adição da constante deve ser armazenado na própria matriz.

* 1. Faça um programa que leia duas matrizes *A* e *B* de tamanho 3 x 3 e calcule *C* = *A* ∗ *B*.
  2. Faça um programa que leia uma matriz *A* de tamanho 3 x 3 e calcule *B* = *A*2.
  3. Na matriz de 20x20 abaixo, quatro números ao longo de uma linha diagonal foram marcadas em negrito. O produto desses números é 26 \* 63\* 78 \* 14 = 1788696.

08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08

49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00

81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65

52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91

22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80

24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50

32 98 81 28 64 23 67 10 **26** 38 40 67 59 54 70 66 18 38 64 70

67 26 20 68 02 62 12 20 95 **63** 94 39 63 08 40 91 66 49 94 21

24 55 58 05 66 73 99 26 97 17 **78** 78 96 83 14 88 34 89 63 72

21 36 23 09 75 00 76 44 20 45 35 **14** 00 61 33 97 34 31 33 95

78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92

16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57

86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58

19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40

04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66

88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69

04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36

20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16

20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54

01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48

Qual é o maior produto de quatro números adjacentes em qualquer direção (cima, baixo, esquerda, direita, ou na diagonal) na matriz de 20x20?

* 1. Faça um programa para determinar a próxima jogada em um Jogo da Velha. Assumir que o tabuleiro é representado por uma matriz de 3x3, onde cada posição representa uma das casas do tabuleiro. A matriz pode conter os seguintes valores: -1, 0, 1 representando respectivamente uma casa contendo uma peça minha (-1), uma casa vazia do tabuleiro (0), e uma casa contendo uma peça do meu oponente (1).

Exemplo:

| -1 | 1 | 1 |
| --- | --- | --- |
| -1 | -1 | 0 |
| 0 | 1 | 0 |