FUNDAÇÃO UNIVERSIDADE FEDERAL DE RONDÔNIA
UNIR

NÚCLEO DE TECNOLOGIA

DEPARTAMENTO ACADÊMICO DE CIÊNCIA DA COMPUTAÇÃO

BACHARELADO E LICENCIATURA EM COMPUTAÇÃO

PROGRAMAÇÃO I

(implementação: Linguagem C)

Profa. Dra. Liliane Jacon

2018

Conteúdo Programático

Conceito de Algoritmos, Programas, Linguagens de Programação. Sintaxe e Semântica. Resolução de problemas e desenvolvimento de algoritmos: análise e solução de problemas, representação e documentação. Estruturas de programas: decisão e repetição. Tipos de dados simples. Modularização de programas: funções e passagem de parâmetros. Tipos de dados compostos: vetores, matrizes, cadeias de caracteres, registros (STRUCT). Estruturas de dados homogêneas e heterogêneas. Arquivo Texto. Programação em linguagem C.

BIBLIOGRAFIA BÁSICA:

FORBELLONE, A. L. VILLAR; EBERSPACHER, H. FREDERICO; **Lógica de Programação** - 3ª Edição. São Paulo: Pearson, 2005.

PAES, Rodrigo de Barros. **Introdução à programação com a linguagem C**.ed.Novatec. São Paulo, 2016.

MIZRAHI, V. V., Treinamento em Linguagem C, 2ª Ed., Pearson Prentice Hall. São Paulo, 2008.

Critério de Avaliação

1ª. nota = 1ª. Prova Bimestral + trabalhos práticos
2ª. nota = 2ª. Prova Bimiestral + trabalhos práticos
3ª. nota = prova substitutiva/repositiva/exame

Média final = (Σ 2 maiores notas) ÷ 2

Ambiente de Programação (software utilizado nos laboratórios) DEV C++

INDICE

1. CONCEITOS BÁSICOS

Algoritmo: Seqüência de passos que visam atingir um determinado objetivo. É uma redação que deve descrever de forma lógica e sem ambigüidades, os passos (ações) a serem seguidos para se resolver um problema específico que tenha um comportamento padrão em sua solução. Exemplo: receita de bolo.

Aplicativos: São desenvolvidos para resolver problemas específicos dos usuários. A aplicação é restrita ao escopo de um determinado problema. Os aplicativos podem ser de natureza técnica, científica, administrativa. Normalmente programas aplicativos são desenvolvidos a partir de um processo de análise com participação de analistas e usuários que possuem conhecimento do assunto. Exemplo: controle de locação de fitas e DVD´s de uma locadora.

Programa: Um programa é um conjunto de instruções que quando executadas produzem um resultado com um desempenho desejado, utilizando-se de estruturas de dados e controle para manipulação adequada de informações. Em resumo, um programa é um Algoritmo escrito em uma linguagem computacional.

Linguagens de Programação: São softwares que permitem o desenvolvimento de programas. Possuem um poder de criação ilimitado, desde jogos, editores de texto, sistemas empresariais até sistemas operacionais. Existem várias linguagens de programação, cada uma com suas características próprias. Exemplo de linguagens computacionais: C, C++, C#, Java, Pascal, Delphi, Ruby, Python.

Sintaxe x Semântica: Numa linguagem, a sintaxe dita as regras de como as frases devem ser construídas corretamente (verbo, sujeito, ...). A semântica se preocupa com o significado da frase construída (entendimento).

Exemplo: “Descoloridas idéias verdes sonham furiosamente”

Sintaxe correta, mas a semântica está confusa!

Observação: A linguagem computacional utilizada nesta apostila será a linguagem C. Isto significa que na resolução de algoritmos utilizando esta linguagem, temos que observar as regras de sintaxe mas também nos preocuparmos com a semântica (em qual sequência as ações devem ser realizadas para atingir o objetivo proposto no algoritmo)

Compilador: Traduz comandos das linguagens de programação e armazena a tradução na memória do computador na forma de um conjunto de instruções na linguagem de máquina, e só depois executa todas essas instruções, gerando-se assim, um arquivo executável.

Interpretador: Programa que reconhece um conjunto de comandos e controla adequadamente o processador, tela e dispositivos de armazenamento. Em resumo, interpreta os comandos “passo-a-passo” na medida da necessidade.

Lógica de Programação: A Lógica de Programação é a técnica de encadear pensamentos para atingir determinado objetivo. Sendo, extremamente, necessária para pessoas que desejam trabalhar com desenvolvimento de sistemas e programas, ela permite definir a seqüência lógica para um programa.

Técnicas de Programação: Formas de se gerar programas, tendo como tipos principais a Programação Estruturada e a Programação Orientada a Eventos e Objetos.

Memória: A Memória do computador é o local onde ficam armazenados os dados e instruções. Ela é organizada em Endereços de Memória, identificados por um código numérico que possibilita a identificação e acesso ao conteúdo de cada endereço. O conteúdo da memória se apaga cada vez que o computador é desligado. Para armazenar os dados e programas é preciso utilizar Dispositivos de Armazenamento: discos rígidos (HD), CD's, HD externos, discos de Zip Drives, mini disks, etc.

Variáveis: Variáveis são endereços de memória destinados a armazenar informações temporariamente.

2. CONSTRUÇÃO DE ALGORITMOS

Sugestões para Construção de Algoritmos

i. Ler atentamente o enunciado;
ii. Retirar do enunciado a relação das entradas de dados;
iii. Retirar do enunciado a relação das saídas de dados;
iv. Determinar o que deve ser feito para transformar as entradas determinadas nas saídas especificadas; (Etapa em que, realmente, se constrói o algoritmo, a partir de alguns requisitos especificados.)
v. Construir o algoritmo;
vi. Executar o algoritmo;
(Implica executar todas as ações descritas seguindo o fluxo de execução estabelecido, verificando se os resultados obtidos correspondem ao esperado quando da montagem do algoritmo, detectando então algum possível erro no desenvolvimento deste. Essa atividade é conhecida por ***"TESTE DE MESA"***.

Exemplo:

Construa um algoritmo (textualmente e em fluxograma) que, calcule a área de um triângulo, onde são dados *Base* e *Altura.*

A fórmula que efetua o cálculo é: $\text{Área} = \frac{Base * Altura}{2}$

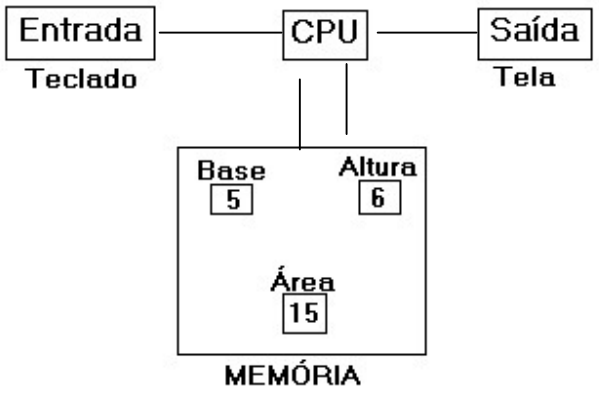
Argumentos de Entrada (AE): base, altura
Argumentos de Saída (AS): área

Ações:

Conheça um valor e atribua ao argumento base
Conheça um valor e atribua ao argumento altura
*Calcule (base*altura/2) e atribua o resultado ao argumento área*
Informe o resultado do argumento área

Fim das ações

Simulação:



Exercícios de Fixação
1:-) Elaborar um algoritmo que informe o maior número entre dois números conhecidos pelo Usuário.

2:-) Construir um algoritmo para calcular a média de dois valores inteiros e positivos, previamente conhecidos.

3. DESENVOLVIMENTO DE PROGRAMAS EM LINGUAGEM C

Linguagem C : é uma linguagem de programação estruturada de uso genérico. Ela contem certas características que permitem que a linguagem seja usada em um nível mais baixo, cobrindo, assim, uma lacuna entre a linguagem de máquina e as linguagens de alto nível.

A linguagem C foi desenvolvida por Dennis Ritchie nos anos 70. Em 1978 Brian Kernighan e Ritchie publicaram uma descrição da linguagem. Em 1984, Bjarne Stroustrup apresentou o C++ (orientada a objetos).

Adicionais

- Comentários no programa são colocados entre /* e */ não sendo considerados na compilação
- Comentário numa única linha (pode ser realizado através de 2 barras invertidas \\.)
- Função getchar(): Armazena a entrada até que a tecla ENTER seja pressionada

3.1 DECLARAÇÃO DAS INFORMAÇÕES UTILIZADAS NUM PROGRAMA EM C

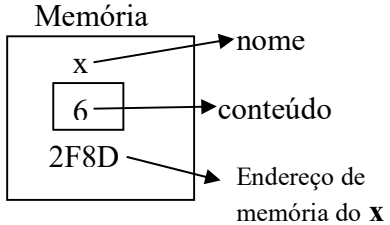
VARIÁVEL: Um dado é classificado como variável, quando tem a possibilidade de ser alterado em algum instante no decorrer do tempo. Ex; o peso de uma pessoa, o valor do dólar (**Utilize sempre identificadores de variáveis em letras minúsculas**)
FORMAÇÃO DE IDENTIFICADORES: Os nomes de informação de caráter variável são os identificadores, os quais devem acompanhar as seguintes regras de formação:

- Devem começar por um caractere alfabético;
- Podem ser seguidos por mais caracteres alfabéticos ou numéricos;
- Não devem ser usados caracteres especiais.

Exemplos:

- Identificadores válidos: Alpha, X, B5, Notas, Médias, INPS
- Identificadores não válidos: 5X, e(13), A:B, Nota/2

Obs: a ling.C, normalmente, reconhece até 31 caracteres



Válidos	Não-válidos
x	salário bruto
total	@mail
soma	taxa%
salario	1x
salarioBruto	Salário-hora
salarioHora	nome do funcionário

Identificadores de dados estão associados algumas características:

- nome do identificador (sempre utilize nomes sugestivos)
- tipo (tipo de informação armazenada: texto, número, etc)
- valor armazenado (conteúdo)

OBS: na linguagem C, identificadores com **letras minúsculas** são diferentes de identificadores com letras maiúsculas. Ex: **T**otal é diferente de **t**otal.

DECLARAÇÃO DE VARIÁVEIS: Nos computadores, as informações variáveis são guardadas na "memória". Como existem muitas variáveis na memória, estas devem ser diferenciadas através dos identificadores (etiquetas). Cada variável, no entanto, pode guardar apenas um dado de cada vez, sendo sempre de mesmo tipo primitivo. Exemplos:

```
int x;
char resp;
float abc, peso;
bool ok, achou;
char nome[20];        // exemplo:
        printf("\nEntre com o nome: ");  scanf("%s",&nome);
```

TIPO	FAIXA ESCALAR
int	2^{-15} a 2^{+15}, ou seja, -32768 a + 32768
unsigned int	Mesmo tamanho que int sem negativo (0 até 65.536)
long	2^{-31} a 2^{+31} ou seja, -2.147.483.648 a +2.147.483.648
unsigned long	Mesmo tamanho que long sem negativos 0 até +4.294.967.298
float	$3,4^{-38}$ até $3,4^{+38}$
double	$1,7^{-308}$ a $1,7^{+308}$
bool	Variável booleana (pode ser true ou false) V ou F Precisa declarar no início do programa #include <stdbool.h>
char	Um único caracter (letra, símbolo ou pontuação) Precisa declarar no início do programa #include <string.h>

As diferenças entre tipos de dados numéricos e caracteres estão na forma de armazenamento e nas operações legais que podem manipular os dados. Exemplo: operações matemáticas podem ser feitas com dados numéricos mas não podem ser feitas com dados caracteres.

3.2 OPERADORES E EXPRESSÕES COMPUTACIONAIS

Relacionais
> maior
>= maior igual
menor
<= menor igual
== igual
!= não igual ou diferente

Lógicos
&& and e
|| or ou
! not não

Aritméticos
% significa o resto de uma divisão inteira
* multiplicação
/ divisão
+ soma
- subtração

Operadores Aritméticos / exemplos:

Expressão Matemática	**Expressão Computacional**
2.a+b	2*a + b
$a \cdot x^2$	a * pow(x,2)
a/2 + b	a/2 + b
$\frac{a + b}{2}$	(a + b) / 2

Prioridade entre os operadores

Operadores	**Ordem de Prioridade**
Parentêses + internos	1ª. parcela a ser resolvida
Funções matemáticas	2ª.
Operadores unários (- ++ -- ! -)	3ª.
Multiplicação divisão e resto	4ª.
Adição e subtração	5ª.

Obs1: Quando houver empate entre 2 ou mais parcelas, a prioridade passa a ser a parcela mais a esquerda. Exemplo: A + B - C

Obs2: Somente o uso de parênteses pode quebrar a ordem de prioridade. Se na mesma expressão, existirem parênteses aninhados (um dentro do outro), a 1ª. prioridade é a do parênteses mais interno. Exemplo: A / (B+C) – (X * (B+C))

Obs3: Exemplos de funções matemáticas:

abs(i) retorna o valor absoluto de i	pow(d1,d2) retorna d1 elevado à potência d2
cos(d) retorna o cosseno de d	rand() retorna um inteiro positivo
exp(d) eleva *e* à potência d (*e*=2.7182818 é a base do sistema natural (naperiano)	sin(d) retorna o seno de d
ceil(d) arredonda para cima –próximo inteiro	sqrt(d) retorna a raiz quadrada de d
tolower(c) converte a letra p/minúscula	tan(d) retorna a tangente de d
toupper(c) converte a letra p/maiúscula	toascii(c) converte o valor do argumento para ASCII

Exercícios

1) Escrever as seguintes expressões algébricas como computacionais

r^2	a + b/c	
$\frac{1}{1/r + 1/s + 1/t} =$	$a + \frac{b}{c+d}$	$\frac{2}{2 - 5y}$.
$(x + y)^3$	$X + y^3$	1/3 . b. h
x^{i+j}	$B^2 - 4ac$	$(x/y)^{y-1}$
$(a-b) \cdot x^3$	$\frac{-b + raiz(delta)}{2^a}$	$\frac{(a+b)^{0.5}}{c^2 - 2a}$

2) Escrever as seguintes expressões computacionais como algébricas

a+b/c +pow(d,e) * f - g	A*b*c / d*e*f	pow(a,m) * pow(b,n) / pow(d,j)
2,1 * pow(x,2) + 3,6 *x	-b + sqrt(delta) / 2 * a	(e-3)/ (z-2) ** 0.5
a* b/c *d / e * f	a * (b-c) / (c+e) * f	

Operadores Relacionais (>, <, >=, <= , ==, !=)
São utilizados para a comparação entre 2 objetos do mesmo tipo. O resultado será sempre um valor do tipo booleano (verdadeiro ou falso)
Exercício: Calcule cada expressão a seguir (se verdadeiro ou falso)

2*4 == 24/3
15 % 4 < 19 % 6
2 + 8 % 7 >= 3 * 6 – 15

Operadores Lógicos (or || && e ! not)
Considere a tabela verdade com duas variáveis booleanas A e B.

A	B	A && B	A || B	A xou B	!A	!B
V	V	V	V	F	F	F
V	F	F	V	V	F	V
F	V	F	V	V	V	F
F	F	F	F	F	V	V

Exercício: Calcule cada expressão a seguir

(2 < 5) && (15/3 == 5)
(2 < 5) || (15/3 == 5)
(2 < 5) xou (15/3 == 5)

3.3 COMANDOS DE ENTRADA E SAÍDA

Algoritmos precisam ser "alimentados" com dados para que possam efetuar as operações e cálculos necessários para alcançar o resultado desejado. Com esta finalidade usam-se os comandos de entrada e saída.
Para utilizar os comandos PRINTF e SCANF é necessário declarar: **#include <stdio.h>**
Saída de Dados
Para que o algoritmo possa mostrar os dados que calculou, como resposta ao problema que resolveu, adotaremos um comando de saída denominado "PRINTF", o qual tem a finalidade de exibir o conteúdo da variável identificada.

Função printf()

Sintaxe:
printf("expressão de controle",argumentos);

É uma função de I/O, que permite escrever no dispositivo padrão (tela).
A expressão de controle pode conter caracteres que serão exibidos na tela e os códigos de formatação que indicam o formato em que os argumentos devem ser impressos. Cada argumento deve ser separado por vírgula.

\n	nova linha	%c	caractere simples
\t	tab	%d	decimal
\b	retrocesso	%e	notação científica
\"	aspas	%f	ponto flutuante
\\	barra	%o	octal
\f	salta formulário	%s	cadeia de caracteres
\0	nulo	%u	decimal sem sinal
		%x	hexadecimal

Ex:
```
main()
	{
	printf("Este é o numero dois: %d",2);
	printf("%s está a  %d milhões de milhas\ndo sol","Vênus",67);
	}
```

Entrada de Dados

Para que o algoritmo possa receber os dados que necessita adotaremos um comando de entrada denominado “SCANF”, o qual tem a finalidade de atribuir o dado a ser fornecido à variável identificada.

Também é uma função de I/O implementada em todos compiladores C. Ela é o complemento de printf() e nos permite ler dados formatados da entrada padrão (teclado). Sua sintaxe é similar a printf().

```
scanf("expressão de controle", argumentos);
```

A lista de argumentos deve consistir nos endereços das variáveis. operador para tipos básicos chamado operador de endereço e refe símbolo "&" que retorna o endereço do operando.

Operador de endereço &:

A memória do computador é dividida em bytes, e são numerad limite da memória. Estas posições são chamadas de endereços. Toda uma certa localização na memória, e seu endereço é o primeiro byte ocu

Ex:

```
main()
        {
        int num;
        printf("Digite um número: ");
        scanf("%d",&num);
```

3.4 COMANDO DE ATRIBUIÇÃO

Um comando de atribuição permite-nos fornecer um valor a uma variável, em que o tipo de dado deve ser compatível com o tipo da variável. Exemplos:

```
int a,b;
float x;
a  =  b;
x  = (8 + 13) /  5 ;
delta = pow(b,2) – 4 * a * c;
```

OBS: - Nos comandos em que o valor a ser atribuído à variável é representado por uma expressão aritmética ou lógica, estas devem ser resolvidas em primeiro lugar, para que depois o resultado possa ser armazenado na variável.

OBS2: Para utilizar funções matemáticas pré-definidas da linguagem C, tais como **pow** e **sqrt**, você precisa declarar no início do seu programa:

#include <math.h>

3.5 EXERCÍCIOS

1:-) Construir um algoritmo para calcular as raízes de uma equação do 2º grau, sendo que os valores dos coeficientes *A, B,* e *C*, devem ser conhecidos inicialmente.

- $\Delta=0$ ➔ Raízes iguais
- $\Delta<0$ ➔ Não há raízes reais
- $\Delta>0$ ➔ Raízes diferentes. Sugestão de coeficientes na execução: a=2 , b=3 e c=-2

2) Faça um algoritmo que calcule o diâmetro, a área e a circunferência de um círculo, sabendo que o único dado disponível é o seu raio.

Diâmetro = 2 * raio	Área = PI * raio^2	Circunferência = 2 * PI * raio

3) Fazer um algoritmo que troque os valores de 3 objetos A, B e C

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <iostream.h>
void main()
{
   clrscr();
   printf( "\nExibe os valores de A B e C");
   int a = 10;
   int b = 20;
   int c = 30;
   printf("\n A = %d", a);
   printf( "\n B = %d" , b);
   printf("\n C = %d" , c);
   printf( "\n\n Apos a troca dos valores tem-se:");
   int temp = a;
   a = b;
   b = c;
   c = temp;
   printf("\n A = %d", a);
   printf("\n B = %d" ,b);
   printf("\n C = %d",c); getchar();
}
```

4) Construa um algoritmo que calcule a média aritmética entre 4 notas fornecidas pelo usuário.

5) Fornecidos os valores dos catetos de um triangulo retangulo, calcular e exibir a hipotenusa, o perímetro e a sua área.

6) Fornecido o valor de T, calcule o valor de W sabendo que:

W = 2T + 4X – 3Y, onde X = 2T – 4 e Y = T/2 + 4

7) Um funcionário de uma empresa recebeu um abono de 20% sobre o seu salário atual e mais uma comissão de $1000,00 e sobre esse total, recebeu um aumento de 35%. Calcule o valor do abono, do aumento e do salário final após estes reajustes, ao ser fornecido o seu salário atual. Veja resolução do algoritmo na linguagem C

```
void main() {
    char nome[20];
    float salatual,abono,comissao,aumento,salbruto;
    printf( "\nFolha de Pagamento - 1 funcionario";
    printf( "\n Entre com o nome do funcionario = ";
    scanf("%s",&nome);
    printf( "\n Entre com o salario atual = ";
    scanf("%f",&salatual);
    abono = salatual * 0.20;
    comissao = 1000.00;
    aumento = (salatual + abono + comissao) * 0.35;
    salbruto = salatual + aumento;
    printf( "\nAbono = %f", abono);
    printf("\nComissao = %f" ,comissão);
    printf( "\nAumento = %f", aumento);
    printf( "\nSalario Bruto = %f", salbruto);
    getchar();
}
```

8) Elaborar um algoritmo que faça a conversão de graus Fahrenheit para Celsius. O algoritmo deve ler um valor em graus Fahrenheit e transformá-lo através da fórmula:

$$C = \frac{(F - 32) * 5}{9}$$

9) Ao serem fornecidos um valor a ser pago e uma taxa de juros de multa, calcule e informe o valor da multa e o total a ser pago.

10) Um comerciante deseja saber qual é o lucro percentual que ele está tendo com a venda de mercadorias. Calcule o lucro percentual de uma mercadoria ao serem fornecidos o preço de compra e o preço de venda da mesma.

4 ESTRUTURAS DE PROGRAMAÇÃO

4.1 ESTRUTURAS DE SELEÇÃO

Em alguns programas existem comandos mutuamente exclusivos, isto é, se um comando for executado, os demais não serão. Quando este for o caso, um comando seletivo é o mais indicado, e este comando seletivo tem a seguinte forma:

- **Seleção simples:** permite determinar se um certo conjunto de instruções deve ou não ser executado.

SE (condição)
(comandos);
	If (condição)
	{
	(comandos);
	}

- **Seleção composta**

SE (condição)
(comandos);
senão
(comandos);
	If (condição)
	{
	(comandos);
	}
	else {
	(comandos);
	}

Exercícios: Resolva os 3 exercícios a seguir utilizando a seleção simples (IF)

1) Faça um algoritmo que receba três notas de um aluno, calcule a média aritmética entre as três notas e exiba mensagem de Aprovado ou Reprovado, considerando a média de aprovação 6,0.

2) Dado um número inteiro, calcule o triplo do mesmo caso ele seja positivo ou nulo, ou o dobro caso ele seja negativo.

3) Fazer um algoritmo que ao ser fornecido o salário de um trabalhador, seja calculado o IR sabendo que se o mesmo for inferior a $1000,00 o valor da alíquota é de 0%, caso contrário será de 10%

OBS: refazer os exercícios acima utilizando a seleção composta (IF ...ELSE)

4) faça um programa que leia a idade (inteiro) e informe sua classe eleitoral

- Não eleitor (abaixo de 16 anos)
- Eleitor obrigatório (>=18 anos ou <= 65 anos)
- Eleitor facultativo (entre 16 e 18 anos e > 65 anos)

Operador Ternário

Este comando nos dá uma maneira compacta de expressar uma simples instrução IF ELSE.

```
Sintaxe:
    condição?expressão1:expressão2
E uma maneira compacta de expressar if-else.
Ex:
main()
        {
        int x,y,max;
        printf("Entre com dois números: ");
        scanf(%d,%d,&x,&y);
        max=(x>y)?1:0;
        printf("max= %d\n",max);
        }
```

Estrutura de Múltipla Escolha (switch)

Algumas vezes, o programa necessita escolher uma entre várias alternativas. É possível utilizar IF ELSE, mas não fica elegante, pois um if tem que ficar dentro do outro (encadeamento de if´s), dificultando o entendimento do programa. O comando switch tem maior flexibilidade e um formato limpo e claro.

Veja exemplo a seguir:

´Uma variável é testada sucessivamente contra uma lista de vari
de caracteres. Depois de encontrar uma coincidência, o comando
comandos é executado.

Se nenhuma coincidência for encontrada o comando default se
default é opcional. A seqüência de comandos é executada até que o
seja encontrado.

```
Ex:
main()
        {
        char x;
        printf("1. inclusão\n");
        printf("2. alteração\n");
        printf("3. exclusão\n");
        printf("  Digite sua opção:");
        x=getchar();
        switch(x)
                {
                case '1':
                        printf("escolheu inclusão\n");
                        break;
                case '2':
                        printf("escolheu alteração\n");
                        break;
```

Continuação dos exercícios utilizando a estrutura de seleção SE

5) Dados dois números inteiros e distintos, construa um algoritmo que seja capaz de definir qual é o maior elemento.

6) Após o conhecimento de um numerador e um denominador, construa um algoritmo para fornecer o resto desta divisão (sem utilizar a função %).

7) Dados dois números inteiros, construa um algoritmo que seja capaz de definir se estes são iguais, e caso isso não ocorrer, qual é o menor elemento.

8) Construa um algoritmo que seja capaz de definir qual o maior elemento entre três números fornecidos pelo usuário (A, B e C)

9) Dados 3 valores A,B.,C verificar se eles podem ser os comprimentos dos lados de um triangulo e, se forem, verificar se compõem um triangulo eqüilátero, isósceles, ou escaleno. Informar se não compuserem nenhum triangulo.
Para verificar se é triangulo: cada lado é menor do que a soma dos outros dois lados
(A < B+C) e (B < A+C) e (C < A+B) ➔ se isto for verdade, então é triangulo
se três lados iguais = triangulo eqüilátero
se dois lados iguais = triangulo isósceles
se três lados diferentes = escaleno

10) Conheça um valor inteiro. Informe se o valor fornecido é par ou ímpar.

11) Construa um algoritmo que, tendo como dados de entrada dois pontos quaisquer do plano P(x1,y1) e Q(x2,y2), imprima a distância entre eles.
Distância = raiz quadrada **(** $(x2 – x1)^2 + (y2-y1)^2$ **)**

12) Elabore um algoritmo que, dada a idade de um nadador, classifique-o em uma das segintes categorias:
infantil A = 5-7 anos
infantil B=8 – 10 anos
juvenil A = 11 – 13 anos
juvenil B = 14 – 17 anos
senior = maiores de 18 anos

13) Ler o sexo (caracter F ou M) e a altura (float)
Calcule e exiba o IMC (índice de massa corporal) IMC = (peso/altura)
IMC < 18,4 → muito magro
18,5 < IMC < 24,9 → ideal
25,0 < IMC < 29,9 → sobrepeso
30,0 < IMC < 34,9 → obeso grau 1
35,0 < IMC < 39,9 → obeso grau 2
IMC >= 40 obesidade mórbida

14) Ler um número e verificar se ele é divisível por 3.
Fazer novamente o algoritmo, mas agora verifique se é divisível por 5.

PROJETO

O DETRAN resolveu instalar um radar no caminho da sua casa e a sua faculdade com o objetivo de limitar a velocidade dos carros, mas resolveram inovar e a velocidade pode ser alterada diariamente. Por ex, nos feriados, a velocidade máxima é de 80 km/h. Já durante a semana, é comum a velocidade máxima seja de 60 km/h.

Para isto, placas digitais informam os motoristas qual é a velocidade máxima naquele momento.
A Tabela de penalizações para quem ultrapassar é:

- Velocidade de até 20% superior ao permitido: multa de $85,13 e 4 pontos na carteira de habilitação;
- Velocidade maior que 20% e até 50% acima do permitido: multa de $127,69 e 5 pontos na carteira de habilitação;
- Velocidade acima de 50% acima do permitido: multa de $574,62 e 7 pontos na carteira, apreensão da carteira e suspensão do direito de dirigir.

Ler: velocidade máxima da via, a velocidade do carro, e deve ser calculado o valor da multa e quantos pontos na CNH o motorista perdeu, caso ele tenha ultrapassado o limite.

4.2 ESTRUTURAS DE CONTROLE DE ITERAÇÕES

Os comandos FOR, WHILE e DO...WHILE permitem que um bloco de comandos seja executado repetidas vezes. Estas estruturas são utilizadas quando se deseja executar um mesmo trecho do programa repetidas vezes.

ITERAÇÕES		
Comando	**Descrição do comando**	**Exemplo**

repetição com variável de controle **FOR**	imprime o nome 10 vezes:
	
	for (int i=0; i<10; i++)
	print("\n Liliane");
		outro exemplo (limpa tela):
		for (int linha=1;linha<=24;linha++)
		{
		for(int coluna=1;coluna<=40;coluna++)
		print("-");
		print("\n");
		}

Repetição com teste no início **WHILE**	while (condição) comandos;
	- ele é executado desde que a condição seja verdadeira
	- testa a condição antes de entrar no laço
		int i = 0;
		while (**i < 10**)
		{
		printf("numero i = %d" , i);
		i++;
		}
		Executa enquanto esta condição for verdade

repetição com teste no final **DO...WHILE**	do
	{
	comandos;
	}
	while (condição);
		do
		{
		// menu de opções de 1 a 3
		print(" 1 - opção cadastro");
		print(" 2 - opção altera");
		print(" 3 - opção sair");
		scanf("%d",&opção);
		} while (opcao != 3);

1° caso) Repetição com variável de controle (FOR):

```
for  (int  i=0; i < 10 ; i++)  {    // repete 10 vezes o trecho abaixo
    comandos;
}
```

```
#include <stdio.h>#include <stdlib.h>
#include <iostream.h>#include <math.h>
 main( )
{
     float soma =0.,r;
     int i, j; j=1;
     print( "\n Calcula a serie item A ");
     for (i=1;i<51;i++)
    {
        r = j / i;
        soma = soma + r;
        j=j+2;
    }
     printf("\n resultado serie = %f", soma);
     print("\n\n");
     system("PAUSE");
     return 0;
}
```

Calcule o valor do S para cada uma das séries abaixo:

a) $S = \frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \ldots + \frac{99}{50} = 95{,}50$

b) $S = \frac{2^1}{50} + \frac{2^2}{49} + \frac{2^3}{48} + \ldots + \frac{2^{50}}{1}$

c) $S = \frac{37 \times 38}{1} + \frac{36 \times 37}{2} + \frac{35 \times 36}{3} + \ldots + \frac{1 \times 2}{37} = 4080{,}75$

d) $S = \frac{1}{1} + \frac{2}{4} + \frac{3}{9} + \ldots + \frac{10}{100} = 2{,}93$

Calcule o valor das seguintes séries (utilize os 20 primeiros termos):

a) $S = \frac{1000}{1} - \frac{997}{2} + \frac{994}{3} - \ldots = 670{,}78$

b) $S = \frac{485}{10} - \frac{475}{11} + \frac{465}{12} - \ldots$

EXERCÌCIOS DE FOR

1º) Escreva um programa que imprima um triangulo isósceles de altura e largura n. Por ex, se n for igual a 3, você deve exibir

```
*
**
***
```

2º) Escreva um programa que exiba um triangulo de altura 2n-1 e largura n
Por ex, se n=4, você exibe largura=4 e altura 2n-1=7

```
*
**
***
****
***
**
*
```

3º) Escreva um programa que exiba um triangulo com altura n e largura 2n-1
Por ex, se n=6, então largura=6 e altura 2n-1 = 12-1 = 11

```
     *
    ***
   *****
  *******
 *********
***********
```

2º.caso) Repetição com Teste no Início (WHILE)

while (condição) comandos;
- ele é executado desde que a condição seja verdadeira
- testa a condição antes de entrar no laço

Exercícios propostos

a) Exibir seu nome 3 vezes na tela
b) Exibir seu nome 50 vezes na tela
c) Exibir a tabuada do número 7 na tela
d) Ler um número inteiro (entre 1 e 9) e exibir a tabuada.
e) Ler 10 número inteiros. Exibir:
    a. Quantos são positivos
    b. Quantos são negativos
    c. Quantos são nulos (igual a zero)
    d. Quantos são ímpares
    e. Quantos são pares
f) Refazer o exercício anterior, mas a quantidade de números a serem lidos é desconhecida. Finalize o programa quando ler o número 9999.
g) Suponha a existência de 10 alunos. Faça um algoritmo para ler a nota do aluno exibir "Aprovado" (nota >=6) ou "REPROVADO" (nota < 6)

h) Refaça o exercício anterior, mas é desconhecida a quantidade de alunos na sala. Quando ler nota = -1 finalize o programa. Informe ainda:
    - quantos alunos foram aprovados e quantos foram reprovados
    - a média da classe e qual a maior nota lida.

Novos exercícios:

1) Fazer um algoritmo para ler diversos números informados pelo usuário, e após cada leitura exibir se o número é par ou ímpar. Considere que ao fornecer um valor negativo, o usuário deseja encerrar a entrada de dados.

```
#include <stdio.h>   #include <conio.h>    #include <string.h>
#include <stdlib.h>  #include <iostream.h> #include <iomanip.h>
void main()
{   clrscr();
    cout << "\nUsando while - Verificando se o numero eh par ou nao";
    int a,b;
    printf("\n Entre com valor A = ");
    scanf("%d",&a);
    while (a >=0)
    {
      if (a % 2 == 0)
            printf("\n O numero %d eh par",a);
      else
            printf("\n O numero %d eh impar ",a);
      printf("\n Entre com valor A = ");
      scanf("%d",&a);
    }
}
```

2) Imprimir os números de 1 até 100.
3) Ler 30 números inteiros fornecidos pelo usuário, e exibir depois quantos números ímpares foram informados. Informe também a quantidade de números pares.
4) Calcular a soma de diversos números reais informados pelo usuário. A entrada de dados termina com o número –999.
5) Ler diversos números reais e exibir qual foi o maior de todos. O código –1 sinaliza o fim da leitura.
6) Imprimir os quadrados dos números inteiros de 1 a 10.
7) Imprimir os números pares entre 2 e 50.
8) Ler 10 números reais e imprimir o maior deles. Refaça para X números.
9) Ler 100 números e exibir o maior deles. Refaça o exercício para X números.
10) Ler 30 números e exibir a soma dos nros que são divisíveis por 5. Refaça o exercicio para X nros
11) Ler diversos números reais e exibir qual foi a soma. O valor – 999 é o código de fim de entrada.
12) Ler diversos números reais e exibir quantos foram digitados. O valor –1 é código de fim de entrada
13) Ler diversos números e exibir quantos números ímpares foram digitados. Considere o valor –999 como código de fim de entrada.
14) Ler diversos números e informe quantos números 10 foram digitados. O valor 0 é código de fim da entrada.
15) Ler diversos números e exibir qual foi o menor e o maior número informado.
16) O usuário deve informar um número inteiro e o programa deve calcular e informar o fatorial do número digitado pelo usuário.
17) Escreva um programa para ler 15 números inteiros e positivos e depois informar: qual o maior número, qual o menor número e a média dos números lidos.
18) Escreva um programa que informe quais números inteiros que divididos por 11 dão resto igual a 5, no intervalo de 1000 a 1999.
19) Escreva um programa que leia um conjunto N de informações contendo, cada uma delas, a altura e o sexo da pessoa (“M” masculino e “F” feminino) Calcule e mostre:
    - ✓ a maior e a menor altura;
    - ✓ a média das alturas das mulheres
    - ✓ a média das alturas dos homens

20) Escrever um programa completo com o seguinte MENU:
1 – Ler um número e verificar se ele é perfeito ou não
2 – Gerar e informar os 5 primeiros números perfeitos
3 – Informar todos os números perfeitos entre 999 e 3820.
4 - Sair

Um número perfeito é aquele que é igual a soma dos seus divisores.
Ex: 28 = 1+2+4+7+14

21) Faça um programa para que um usuário adivinhe um número informado por outro. Inicialmente, um usuário fornece um número para o programa, e depois o outro usuário deve informar números até acertar (é claro que o segundo usuário não vê o número registrado pelo primeiro). Para ajudar, a cada tentativa o computador deve exibir a mensagem "Seu número é maior do que o meu" ou "Seu número ´´e menor do que o meu". Ao acertar, imprima também quantas tentativas foram necessárias.

22) Faça um algoritmo que exiba os 15 primeiros números de Fibonacci. A seqüência de Fibonacci, começa com 1 e 1, e cada número subseqüente é a soma dos dois anteriores. Veja: 1 1 2 3 5 8 13 ....

23) Existem dois candidatos a uma vaga de presidente de clube, Firmino e Eugência. Feita a eleição, os votos são enviados por um dispositivo de entrada, sendo cada entrada o voto de um eleitor. Cada voto foi codificado da seguinte forma: 1 representa voto de Firmino, 2 em Eugênia, o voto 0 indica Branco e 9 é voto nulo. Deseja-se saber por um programa:
a) o nome do candidato vencedor
b) o número de votos em branco;
c) o número de votos nulos;
d) a quantidade de eleitores que votaram.

Estabeleça um código que termine a entrada de dados, e esqueça os empates.

24) Faça um programa para ler diversos caracteres informados pelo dispositivo de entrada (utilize a diretiva **#include <ctype.h>**). Depois informe:
a) a quantidade total de letras A e Z informados
b) a quantidade de caracteres informados
c) a quantidade de consoantes
d) a maior letra informada (de acordo com a ordem alfabética)
e) a quantidade de pontos de exclamação informados

A condição de parada deve ser quando for lido o caractere # (considere apenas letras maiúsculas no seu programa).

```
#include <ctype.h>
letra = getchar();
if  (isalpha(letra))
      // encontrou letra;
```

25) A prefeitura de uma cidade fez uma pesquisa entre seus habitantes, coletando dados sobre o salário e número de filhos de cada chefe de família. A prefeitura deseja saber:
a) média do salário
b) média do número de filhos
c) maior salário
d) percentual de pessoas com salário de até $800,00

27) Faça um algoritmo para imprimir a quantidade de números primos entre 1 e 1000.

28) Foi realizada uma pesquisa dos carros usados que estão à venda na cidade de Porto Velho. O programa deve ler as seguintes informações para cada carro:
- placa do carro
- tempo de uso (idade)
- cor (A-azul, P-preto, B-branco e O-outros)
- marca G–gm F-ford V-volks e A-fiat

Não se sabe quantos carros estão a venda. Finalizar quando placa do carro for igual a ZERO.
Faça um programa para calcular e exibir:

a) o total de carros à venda
b) a média do tempo de uso dos carros (a média da idade dos carros)
c) contar quantos carros tem menos de 5 anos de uso, que sejam da cor azul e da marca Gm.
d) a porcentagem de carros com mais de 10 anos de uso

```
#include<conio.h>  #include<stdio.h>
#include <iostream.h> #include <stdlib.h>
int main()
{
  system("CLS");
  int placa,tempo=0, carroGm=0, carro=0;
  int carroDez=0;
  float porcarros, media, somaTempo=0;
  char cor, marca;
  cor=' ';   marca=' ';
  printf("\nInforme um numero para a placa do carro:");
  scanf("%d",&placa);
  while(placa!=0)
  {
        printf("\nInforme o tempo de uso do carro:");
        scanf("%d",&tempo);
        printf("\nInforme a cor do carro:A-azul\n P-preto\n B-branco\nO-outros\n");
         cor=getche();
        printf("\nInforme a marca do carro:G-gm\n F-ford\n V-volks\n e A-fiat\n");
         marca=getche();
        carro=carro+1;
        if ((tempo<5)&&(cor=='A')&&(marca=='G'))
                  carroGm=carroGm+1;
         if (tempo>10)
                  carroDez=carroDez+1;
        somaTempo=somaTempo+tempo;
         printf("\nInforme um numero para a placa do carro:");
        scanf("%d",&placa);
  }
  printf("\n\nO total de carros foram:%d",carro);
  media=somaTempo/carro;
  printf("\nA media é de:%8.2f",media);
  printf("\nA quantidade de carros marca Gm, cor azul e < 5 anos são: %d", carroGm);
  porcarros=(carroDez/carro)*100.;
  printf("\nA porcentagem de carros com mais de 10 anos foram:%8.2f",porcarros);
  getch();   cout<<"\n\n";
  system("PAUSE");  return 0;
}
```

29) Foi realizada uma pesquisa de algumas características físicas da população de uma certa região, a qual coletou os seguintes dados referentes a cada habitante para serem analisados:

- ✓ sexo (masculino ou feminino)
- ✓ cor dos olhos (verdes, azuis, castanhos, pretos)
- ✓ idade

Faça um algoritmo que determine e escreva:

- ✓ A maior idade dos habitantes
- ✓ A percentagem de indivíduos do sexo feminino cuja idade está entre 18 e 35 anos inclusive e que tenham olhos verdes.

O final do conjunto de habitantes é reconhecido pelo valor –1 na entrada da idade.

30) Calcule o imposto de renda de um grupo de contribuintes considerando que os dados de cada contribuinte, número do CPF, número de dependentes e renda mensal são valores fornecidos pelo usuário. Para cada contribuinte será feito um desconto de 5% de salário mínimo por dependente. Os valores da alíquota para calculo do imposto são:

Renda Líquida	Alíquota
Até 2 salários mínimos	Isento
2..3 salários mínimos	5%
3..5 salários mínimos	10%
5..7 salários mínimos	15%
acima de 7 salários mínimos	20%

O último valor que não será considerado, terá o CPF igual a zero. Deve ser fornecido o valor atual do salário mínimo.

31) Construa um algoritmo que leia um conjunto de dados contendo altura e sexo (masculino ou feminino) de 50 pessoas e depois calcule e escreva:

- ✓ a maior e a menor altura do grupo
- ✓ a média da altura das mulheres
- ✓ o número de homens e a diferença percentual entre estes e as mulheres

32) Em uma empresa de calçados, as vendas são anotadas de acordo com o código do calçado, de seu número e de seu preço unitário, como no exemplo:

Código: 203

Num	Quant	PreçoUnit	PreçoParcial
35	2	36,00	72,00
34	3	15,00	45,00
...			
0			
Totais	5		117,00

Código 101

Num	Quant	PreçoUnit	PreçoParcial
38	10	10,00	100,00
34	2	14,00	28,00
37	5	15,00	75,00
...			
0			
Totais	17		203,00

Faça um algoritmo que ao conhecer o código do calçado, seu número e seu preço unitário, calcule:

a) o total de vendas de cada código em quantidade e preço
b) o total de todas as vendas em quantidade e preço
c) fornecido um número, calcule a quantidade de calçados deste número entre todos os códigos
d) o código do calçado com menor preço unitário
e) o código do calçado que obteve a maior venda (em valor)
f) do calçado que obteve a maior venda (em valor)

5. FUNÇÕES NA LINGUAGEM C

Funções são estruturas que permitem ao usuário separar seus programas em blocos.

Formato geral

```
tipo_de_Retorno Nome_Função (parâmetros) {
        corpo da função
}
```

Para retornar V (true) ou F (false), Você deve declarar no início do programa:
#include <stdbool.h>

O comando return

Quando se chega numa declaração return, a função é imediatamente encerrada, e se o valor de retorno é informado, a função retorna esse valor. Uma função pode ter mais de uma declaração return.

```
#include <stdbool.h>
bool  Epar (int a)  {
        if (a%2==0)
                return true;
        else
                return false;
}
```

```
float quadrado(int n) {
        return (n * n);
}
```

```
float media(int n1, int n2){
    float m;
    m =  (n1+n2)/2.;
    return m;
}
```

Protótipo de funções

Caso você queira colocar a função APÓS o main, você deve declarar o protótipo da função ANTES do main para o compilador.

```
#include <stdio.h>
float quadrado (float n);
int main()
{
   .....
}
float quadrado(float n) {
    return (n*n);
}
```

O tipo void

Em inglês, *void* significa vazio. Ele permite fazer funções que **não retornam nada** e/ou também funções que não tem parâmetros. Se queremos que a função retorne algo, devemos usar a declaração return. Se não queremos, basta declarar como sendo do tipo void.

```
void mensagem () {
   printf("\n Alo mundo!!!!");
}
int main() {
    mensagem();
}
```

Diferença entre função void e as demais funções que retornam dados

Exemplo: Fazer um programa que utilize uma função que retorne a média entre 3 valores a,b e c.

```
float calculamedia (int a, int b, int c) {
      float media;
      media = (a+b+c)/3.;
      return media;
}
main(){
   int x,y,z;
   printf("\n Entre com o 1º. Valor: "); scanf("%d",&x);
```

```
   printf("\n Entre com o 2º. Valor: "); scanf("%d",&y);
   printf("\n Entre com o 3º. Valor: "); scanf("%d",&z);
   float resultado = calculamedia(x,y,z);
   printf("\n resultado da media calculado = %f",resultado); getchar();
}
```

Refazer o mesmo exercício acima, utilizando uma função void (sem return)

```
void calculamedia (int a, int b, int c) {
      float media;
      media = (a+b+c)/3;
      printf("\n media calculada dentro da função= %f", media);
}
main(){
   int x,y,z;
   printf("\n Entre com o 1º. Valor: "); scanf("%d",&x);
   printf("\n Entre com o 2º. Valor: "); scanf("%d",&y);
   printf("\n Entre com o 3º. Valor: "); scanf("%d",&z);
   calculamedia(x,y,z); //Repare que a média é desconhecida no programa principal
   getchar();
}
```

5.1 ESCOPO DE VARIÁVEIS

Variáveis Locais: essas são aquelas que só tem validade dentro do bloco ao qual são declaradas.

int fatorial(int n) {
int resul=1;
for (int i=2; i<=n; i++)
resul=resul*i;
return resul;
}
Ex: 4! = 24.
	float expo(int n, int m) {
	int **resul=n**;
	for (int i=1;i<=m;i++)
	resul=resul*i;
	return resul;
	}
	Ex: expo(2,5) = 2^5 = 256
		Na função fatorial temos as variáveis locais i e resul.
		Na função expo temos as variáveis locais i e resul.

5.2 PARÂMETROS FORMAIS E REAIS

Parâmetros Formais: Esses são declarados como sendo as entradas de uma função. Você pode alterar o valor de um parâmetro formal, pois essa alteração não terá efeito na variável que foi passada a função. Quando o C passa parâmetros para uma função, *são passadas apenas cópias das variáveis* (exceção: arranjos homogêneos como vetores e matrizes)

Parâmetros Reais: Quando se faz a chamada a uma função, os parâmetros passados são denominados parâmetros reais.

Programa para calcular o fatorial de um Número inteiro:

```
#include <stdlib.h>
#include <stdio.h>
main()
{
     int n;
     print("\nCalculo de Fatorial\n");
     print("\nEntre com um numero: ");
     scanf(“%d”,&n);
     int resul=1;
     for (int i=2; i<=n; i++)
        resul=resul*i;
     printf ("\nFatorial de %d = %d",n,resul);
     print("\n\n");
     system("PAUSE");
}
```

Utilizando FUNÇÃO para calcular o Fatorial de um Número:

```
#include …..
int fatorial(int n)
{
  int resul=1;
  for (int i=2; i<=n; i++)
     resul=resul*i;
  return resul;
}
main()
{
     int n,resul;
     print("\nCalculo de Fatorial\n");
     print("\nEntre com um numero: ");
     scanf(“%d”,&n);
     resul=fatorial(n);
     printf ("\nFatorial de %d = %d",n,resul);
    // ou
   //  printf ("\nFatorial de %d = %d",n,fatorial(n));
     print( "\n\n");
     system("PAUSE");
}
```

Parâmetro formal

Parâmetros reais

Exercício: Faça um programa para calcular

$$c = \frac{n!}{p!\,(n-p)!}$$

Utilize o exercício de fatorial com FUNÇÃO como referência.

```
int fatorial(int n) {
  int resul=1;
  for (int i=2; i<=n; i++)
     resul=resul*i;
  return resul;
}
```

```
int main()  {
      int n,p;
      print("\nCombinação - Uso da função para calcular Fatorial\n";
      print("\nEntre com um numero N: ";scanf(“%d”,&n);
      print("\nEntre com um numero P: "; scanf(“%d”,&p);
      float resul = fatorial(n) / (fatorial(p)*fatorial(n-p));
      printf ("\nCombinação de N e P = %5.2f ",resul);
      print("\n\n";
      system("PAUSE");
      return 0;
 }
```

EXERCICIO

Faça o teste de mesa com o programa abaixo, e informe qual

será o resultado exibido.

```
int calculos( int x, int y, int z) {
   z = x+y+z;
   printf ("\n x = %f", x);
   printf( "\n y = %f", y);
   printf("\n z = %f",z);
   return z;
}

int main() {
    system("cls");
    printf( "\nExibe os valores de A B e C");
    int a = 5;
    int b = 8;
    int c = 3;
    c = calculos(a,b,c);
    printf("\n A = %f", a);
    printf("\n B = %f", b);
    printf("\n C = %f",c);

    a = calculos(7,a+b+c,a);
    printf("\n A = %f", a);
    printf("\n B = %f", b);
    printf("\n C = %f",c);

    c = calculos(a*b, c-a, c);
    printf("\n A = %f", a);
    printf("\n B = %f", b);
    printf("\n C = %f",c);
    system("pause");
    return 0;
}
```

```
X=5
Y=8
Z=16

A=5
B=8
C=16

X=7
Y=29
Z=41

A=41
B=8
C=16

X=328
Y=-25
Z=319

A=41
B=8
C=319
```

5.3 PASSAGEM DE PARÂMETROS

Quando chamamos uma função em C, os parâmetros formais da função copiam os valores dos parâmetros que são passados para a função.

Exemplo:

```
float  sqr (float num)   {
   num = num * num;
   return num;
}
main() {
    float num, resul;
    print("\nEntre com um numero: "); scanf("%f",&num);
    resul = sqr (num);
    printf ("\n Numero original:  %d ", num);
    printf (" "\n Quadrado de num:  %f" ,resul);
}
```

Outro tipo de passagem de parâmetros é conhecido como "chamada por referência".Neste tipo de chamada, os parâmetros formais alterados dentro da função voltam alterando os parâmetros reais (utilizados na ativação da função).

Exemplo:

```
#include <stdio.h>   #include <stdlib.h> #include <iostream.h>

void troca(int &a, int &b) {
  int tmp=b;
  b=
  a;
  a=tmp;
}
```

```
int main( ) {
    int x=1,y=2;
    printf("\n Antes da troca");
    printf("\n x: %d ",x);
    printf("\n y: %d",y);
    troca(x,y);
    printf("\n Depois da troca");
    printf("\n x: %d ",x);
    printf("\n y: %d",y);
    printf("\n\n");
    system("PAUSE");
    return 0;
}
```

Exercícios

1) Faça um programa que possua uma função para calcular a área de um círculo.

   área = π * raio

   OBS: utilize a diretiva #define (repare que não existe o sinal de igual ao usar o define)

   #define PI 3.141592

2) Faça um programa que possua uma função para calcular o quadrado de um valor inteiro.
   Quadrado = sqr (numero)

3) Faça um programa que possua uma função para calcular exponenciação entre dois valores inteiros n e m (n^m)

4) Escreva um programa que utilize uma função que retorne 1 se o argumento é um número ímpar e 0 se o argumento for um número par.

5) Um número primo é qualquer inteiro positivo que é divisível apenas por si próprio e por 1. Escreva uma função que recebe um inteiro positivo e, se este número for primo, retorna 1, caso contrário, retorna 0.

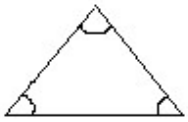
6) A famosa conjetura de Goldbach diz que todo inteiro par maior que 2 é a soma de dois números primos. Testes extensivos foram feitos sem contudo ser encontrado um contra-exemplo. Escreva um programa mostrando que a afirmação é verdadeira para todo número par entre 700 e 1100. O programa deve imprimir cada número e os seus correspondentes primos. Use a função do exercício anterior.

7) Faça um programa contendo uma função para verificar se o dia é válido (está no intervalo entre 1 e 31) e se o mês está entre 1 e 12. Caso esteja incorreto, ler nova data.

8) Faça um programa que utilize uma função para retornar um valor calculado do seguinte modo:

Dentre 4 argumentos, escolhe-se o maior dos dois primeiros argumentos e o maior dos dois últimos e, em seguida, calcula-se a raiz quadrada da soma dos dois maiores, subtraída do produto dos outros dois.

9) No triangulo abaixo, se forem conhecidos os valores dos ângulos ALFA e BETA, e o comprimento do lado C, podem ser determinados os outros valores



$$\frac{\text{sen } \alpha}{a} = \frac{\text{sen } \beta}{b} = \frac{\text{sen } \theta}{c} \qquad \alpha + \beta + \theta = 180^o$$

Faça um programa para:

a) no programa principal (main) ler: ladoC, alfa e beta (todos double);
b) uma função (double) ANGULOS que receba como parâmetros alfa e beta e calcule e retorne o terceiro ângulo restante (teta);
c) uma função (double) CalculaLado, que receba de parâmetros dois ângulos e o valor do lado C, e que calcule e retorne um dos lados faltantes do triangulo;
lembre-se: Math.sin(alfa)
d) no programa principal exibir o ângulo teta e os lados A e B calculados;

10) Faça um programa completo contendo as seguintes funções abaixo:

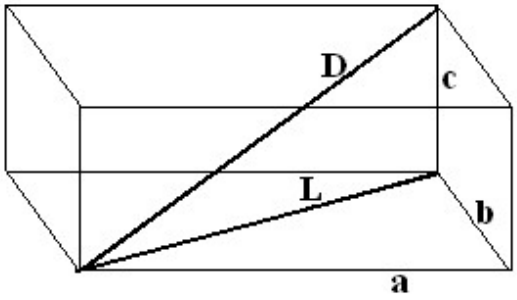
a) função que ao receber como parâmetros um mes e um ano, retorne a quantidade de dias deste mês;
b) função que ao receber um ano, retorne a quantidade de dias deste ano;
c) função que ao receber como parâmetros uma data (dia, mês e ano) retorne a quantidade de dias de 1/1/1901 até esta data;
d) função que ao receber como parâmetros duas datas (dia1,mes1,ano1,dia2,mes2,ano2) retorne a diferença em dias entre estas duas datas.

No principal, faça um menu de opções para cada função acima. Exemplo:

```
1 – Total de Dias do Mês
2 – Total de dias do Ano
3 – Total de Dias desde 1/1/1901
4 – Diferença entre duas datas
```

11- Sejam P(x1,y1) e Q(x2,y2) dois pontos quaisquer do plano. A sua distância é dada por:
Distancia = $\sqrt{(x2 - x1)2 + (y2 - y1)2}$
Faça um programa que contenha uma função para calcular a distância entre os pontos P e Q.

12 – Faça um programa completo que:
Calcule e exiba o valor da diagonal D de um paralelepípedo de lados a, b, c.
Obs: no principal, você só pode ler a, b e c. Os valores L e D devem ser calculados.



L = raiz quadrada ($a^2 + b^2$)
D = raiz quadrada ($L^2 + c^2$)

Sugestão: faça uma função que calcule a hipotenusa de um triangulo retângulo.
float hipotenusa(float catA, float catB)
Lembrando que hipotenusa = raiz quadrada ($catA^2 + catB^2$)

6. CADEIA DE CARACTERES

Em C, string (cadeia de caracteres) não é um tipo de dado formal, como Java ou C#.
String é um vetor do tipo CHAR terminada pelo caractere null (‘\0’).
O último elemento da string é um ‘\0’ (inteiro nulo). Exemplo:

```
char frase[10];
strcpy(frase, “Liliane”);
```

0	1	2	3	4	5	6	
L	i	l	i	a	n	e	\0

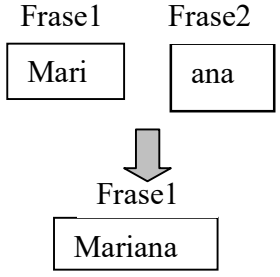
OBS: para utilizar as funções apresentadas é preciso incluir: #include **<string.h>**

Para ler e exibir uma string: Exemplo:

```
char nome[30];
printf(“\nDigite um nome:”);
scanf(“%s”,&nome);
printf (“\n \n Ola sr(a) %s “,nome);
```

Função **strcat** (copia uma string origem para uma string destino). Exemplo:

```
char frase1[10];
char frase2[10];
strcpy(frase1, “Mari”);
strcpy(frase2,”ana”);
strcat(frase1,frase2);
printf("\n frase1= %s",frase1);
printf("\n frase2= %s",frase2);
```



Função **strlen** (retorna o comprimento da string fornecida). O terminador nulo não é contado. Exemplo:

```
char nome[100];
strcpy(nome, "Liliane Jacon");
char frase[40];
int tam = strlen(nome);
```

13

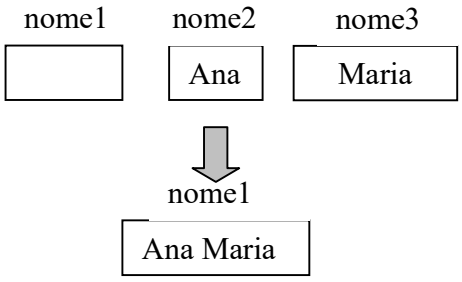
```
printf("\n tamanho do nome %s = %d ",nome,tam);
printf("\nEntre com uma frase: "); scanf("%s",&frase);
printf("\n tamanho da frase %s = %d ",frase,strlen(frase));
```

Função **strcmp**: Compara a string1 com a string2. Se as duas forem idênticas, retorna zero, caso contrário (strings são diferentes) retorna não-zero. Exemplo:

```
char cad1[20], cad2[20];
printf("\nEntre com a cadeia 1: ");   scanf("%s",&cad1);
printf("\nEntre com a cadeia 2: ");  scanf("%s",&cad2);
if (strcmp(cad1,cad2)==0)
        printf("\nDuas cadeias são iguais");
   else  printf("\nDuas cadeias são diferentes");
```

OBS: Não se pode igualar duas strings, por ex: string1 = string2; //errado!!!

Inicializando Strings




```
static char nome1[15]="";
char nome2[10], nome3[15];
strcpy( nome2, "Ana  ");
strcpy( nome3, "Maria");
strcat(nome1,nome2);
strcat(nome1,"  ");
strcat(nome1,nome3);
printf("\n %s tamanho %d",nome1,strlen(nome1));
char frase[20];
printf("\nDigite uma frase grande:");  scanf("%s",&frase);
strcat(nome1,frase);
printf("\n %s tamanho %d",nome1,strlen(nome1));
```

6.1 EXERCÍCIOS UTILIZANDO CADEIA DE CARACTERES

1. Faça um programa que tendo como dados de entrada um texto com no máximo 50 caracteres e uma letra, que seja capaz de calcular a freqüência desta letra no texto.

Frase= mamae mandou eu bater neste daqui, mas como sou teimosa bato nesse daqui
Letra = a
Freqüência da letra a na frase = 9

```
#include <stdio.h>
#include <string.h>
        int main() {
            char frase[50];
            char letra='\0';
            printf("\nDigite uma frase:"); scanf("%s",&frase);
            printf("\nDigite uma letra:"); letra=getche();
            int freq=0;
            for (int i=0; i<=strlen(frase); i++)
                  if (frase[i]==letra) freq=freq+1;
            printf("\nFrequencia da letra %c na frase = %d\n\n",letra,freq);
        }
```

2. Faça um programa que seja capaz de inverter uma cadeia de caracteres. A cadeia deverá conter no máximo 50 caracteres.

```
char frase[50];
char resul[50];
printf("\nDigite uma frase:"); scanf("%s",&frase);
```

```
strcpy(resul,frase);
int j=strlen(frase)-1;
for (int i=0; i<strlen(frase); i++)    {
   resul[j]=frase[i];
   //printf("\nresul= %s i= %d  j= %d\n",resul,i,j);
   j--;
}
printf("\nFrase resultante = %s\n\n",resul);
```

3.Faça um programa para contar quantas palavras tem numa frase. Considere que existe um único espaço entre uma palavra e outra.

Frase= as meninas super poderosas estão de volta Contagem= 7

```
char frase[50];
printf("\nDigite uma frase:"); scanf("%s",&frase);
int cont;
if (frase[0]!=' ') cont=1;
            else cont=0;
for (int i=0;i<strlen(frase);i++)
   if (frase[i]==' ')
      cont=cont+1;
  printf("\n\nTotal de palavras da frase = %d\n\n",cont);
```

4. Faça um programa para substituir por asteriscos todas as palavras de tamanho 3.

Frase = "o rei de Roma riu ao ver o fogo"

Resultado = "o *** de Roma *** ao *** o fogo"

5. Passar para maiúsculo todas as palavras que começam por vogais (A,E,I,O,U)

Frase= ratao comeu ameixa ontem

Resultado = ratao comeu AMEIXA ONTEM

```
char frase[50];
printf("\nDigite uma frase:"); scanf("%s",&frase);
int cont;
for (int i=1;i<strlen(frase);i++)  {
if ((frase[i-1]==' ') && ((frase[i]=='a') ||(frase[i]=='e') ||(frase[i]=='i') ||(frase[i]=='o') || (frase[i]=='u'))) {
      int k=i;
      while ( (frase[k]!=' ') && (k<strlen(frase)))     {
        int x = frase[k] - 32; // 65=A  97=a
        char letra=x;
        frase[k]= letra;
        k++;          }
  }
}
 printf("\nFrase refeita = %s\n\n",frase);
```

6. Faça um programa que tenha como entrada uma cadeia de caracteres e dois valores inteiros (o 1º. indicando uma posição inicial e o 2º. indicando uma posição final da mesma). O programa deverá exibir como resultado a cadeia compreendida entre o 1º. e o 2º. valor,inclusive.

Frase= PADARIA

Num1 = 2 num2 = 4

Resultado = DAR

7. Faça um programa que tenha como entrada uma cadeia de caracteres, e dois valores inteiros (o 1º. indicando uma posição inicial da cadeia e o 2º. indicando uma posição final da mesma). Que substitua na própria cadeia, o que havia nela inicialmente com exceção dos caracteres compreendidos entre o 1º. e o 2º. valor, inclusive. Exemplo:

8. Faça um programa que tenha como entrada duas cadeias de caracteres e um valor inteiro (especificando uma posição na 1ª. cadeia). O algoritmo deve substituir na 1ª. cadeia o que continha nela inicialmente, acrescida com a 2ª. cadeia a partir da posição especificada. Exemplo:

9) Dado uma cadeia de caracteres com uma frase de 50 letras (incluindo brancos) escrever um algoritmo para :

a) contar quantos brancos tem na frase
b) contar quantas vezes aparece uma determinada letra
c) contar quantas vezes aparece a letra "p" precedida pela letra m (ex: sempre)
d) contar quantas vezes aparece uma letra com duas ocorrências consecutivas (exemplo assim, passagem, ferreira)

10) Faça um programa para ler um nome completo (nome e sobrenome).
Converta e exiba o nome lido para o formato de citação bibliográfica (primeiro o sobrenome, uma vírgula e depois o nome).
Exemplo "Liliane Jacon"
Conversão "Jacon, Liliane"

11) É dado um texto que consiste de uma ou mais palavras, separadas cada uma da outra por um ou mais espaços e que termina por zero ou mais espaços seguido de ponto(.)
Faça um algoritmo que:

a) as palavras consecutivas devem ser separadas por um único espaço
b) a última palavra deve ser seguida por ponto
c) se numerarmos as palavras do texto na ordem da esquerda para a direita (0,1,2...) então as palavras de número par devem ser copiadas enquanto as palavras de número ímpar devem ser invertidas.

Ex: aqui__ esta_ um___exemplo.
Resultado: aqui atse um olpmexe.

7. ARRANJOS / VETORES

Vetores são arranjos homogêneos unidimensionais (de 1 dimensão).
Formato Geral

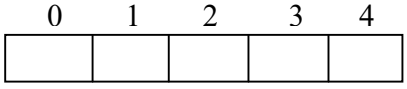
```
tipo nome[tamanho];
```

Quando o C vê uma declaração dessa, ele reserva um espaço na memória para armazenar os elementos dentro do vetor. Ex: **int vetor[5];**

No C, a numeração começa sempre em zero. Isso significa que no comando **int vetor[5],** os dados serão indexados de 0 a 4.

tf = tamanho físico = espaço total reservado na memória

tl = tamanho lógico = espaço ocupado/preenchido



Operações Básicas

float vetor[5];
vetor[0]=8.3;
vetor[1]=10.5;
vetor[2]=vetor[0] * vetor[1];
	int arranjo[8];
	arranjo[ ]={ 17, 4, 9, 13, 5, 8, 13, 21};
	
	arranjo[3]=arranjo[1]+arranjo[2];

O C não verifica se você está respeitando os limites do vetor. Cuidado com a lógica!!
Supor que declarou um vetor de 8 posições, ou seja, com posições de 0 até 7.
Ex: vetor[9] = 25;
// o programa irá compilar, mas ocorrerá um erro ao executar o programa.

```
#include <stdio.h> #include <string.h>
#define tf  5
void leitura(int vetor[ ]) {
    int i;
    for (i=0; i < tf; i++)    {
       printf("\n entre com um numero da posição [%d]: ", i);
       scanf("%d",&vetor[i]);
    }
}
void exibe(int vetor[ ]){
   int i;
   printf("\n\n Os numeros que você digitou foram: ");
   for (i=0; i < tf; i++)
         printf("\n vetor [%d] = %d ", i , vetor[i] );
}

void calculamedia(int vetor[ ]){
    float soma=0;
    float media;
    int i;
    for (i=0; i< tf; i++)
        soma = soma + vetor[i];
    media = soma / tl;
    printf("\n resultados foram soma=%f   media=%f",soma,media);
}

main(){
     int vetor[tf],  opcao=0;

     while (opcao != 4) {
        printf("\n 1 – leitura");
        printf("\n 2 - exibe vetor");
        printf("\n3 – calcula soma e media");
        printf("\n 4 – sair");
        printf("\n Opcao?");
        scanf("%d",&opcao);
        switch(opcao) {
            case 1:  leitura(vetor);
                     break;
            case 2: exibe(vetor);
                    break;
```

Suponha que você digite:
25 35 83 18 22
O vetor ficará desta forma:

0	1	2	3	4
25	35	83	18	22

```
            case 3: calculamedia(vetor);
                    break;
       }//switch
     }//while
}//main
```

7.1 EXERCÍCIOS COM VETORES

1) Faça um programa que seja capaz de conhecer e armazenar TL valores inteiros quaisquer, com capacidade física total de 15 elementos. Calcule e exiba a média aritmética dos elementos do vetor.

2) Refaça o exercício 1. No final, exiba todos os elementos e suas respectivas posições no vetor, somente dos elementos cujos valores são maiores que a média.

3) Faça um programa para ler 2 vetores inteiros de tamanhos idênticos. Calcule e exiba a soma dos vetores num terceiro vetor resultante.

Vetor1

0	1	2	3	4
10	20	30	40	50

Vetor2

0	1	2	3	4
5	8	11	90	-6

Vetor resultante da soma

0	1	2	3	4
15	28	41	130	44

```
#include <stdio.h>
#define tf  5
void leitura(int v[],int tl) {
    for (int i=0;i<tl;i++)      {
       printf("\n vetor[%d]= ",i);
       scanf("%d",&v[i]);
     }
}
void soma(int v1[],int v2[],int vR[],int tl) {
  int i;
  for (int i=0;i<tl;i++)   {
    vR[i]=v1[i]+v2[i];
  }
}
 main() {
     int vetor1[tf],vetor2[tf],vetorR[tf];
     int tl;
     printf("\n entre com o tamanho dos vetor (TL): ");
     scanf("%d",&tl);
     printf("\nVetor 1");       leitura(vetor1,tl);
     printf("\nVetor 2");      leitura(vetor2,tl);
     soma(vetor1,vetor2,vetorR,tl);
     printf("\nO vetor resultante da soma...\n");
     for (int i=0;i<tl;i++)
        printf("\n vetorR[%d]= %d \n",i,vetorR[i]);
     printf("\n\n\n");
     getchar();}
```

4) (função busca) Faça um programa para ler um vetor com tamanho TL (máximo de 10 posições) e um dado elemento (numeroProcurado). Verifique se o elemento está contido no vetor. Caso afirmativo, informe a posição do elemento. Caso negativo, informe que numeroProcurado não se encontra no vetor. Sugestão: retorne posição = -1 caso não encontre o numeroProcurado dentro do vetor.

int posição = BUSCA (vetor, tl, numeroProcurado);

5) Faça um programa para ler 2 vetores e seus respectivos TL´s. Calcule num terceiro vetor a união dos 2 vetores lidos. Exiba o vetor união. OBS: não há elementos repetidos no vetor UNIÃO.

6) (nova inserção) Faça um programa para ler um vetor de TL elementos. O programa deverá inserir um novo elemento em uma determinada posição do vetor. Os elementos devem ser remanejados para a inserção do novo elemento. Veja explicação abaixo:

vetor Inicial

TL inicial=3

0	1	2	3	4
10	20	30		

Inserir o elemento 15 na posição 1

vetor Resultante

TL final=4

0	1	2	3	4
10	15	20	30	

7) (remoção) Leia um vetor e uma posição. Retire do vetor o elemento da posição especificada. Remaneje os elementos do vetor resultado. Exiba o vetor resultado.

Vetor Inicial

TL inicial=5

0	1	2	3	4
10	20	30	40	50

Posiçao = **1** (posição do elemento a ser retirado)

Vetor Resultado

TL final=4

0	1	2	3	4
10	30	40	50	

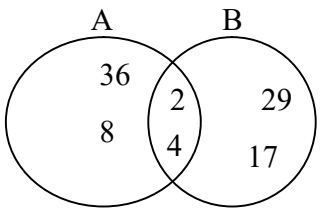
8) (Diferença entre 2 vetores). Leia 2 vetores A e B de 10 posições de inteiros cada um.
Faça uma função para criar um terceiro vetor (vetorC) que armazenará os elementos resultantes da diferença entre os vetores A e B (**A – B**), ou seja, todos os elementos que estão no vetor A mas não estão no vetor B. Para o vetor C não conter elementos do vetor B, você pode (e deve!) usar a função busca. Suponha que as funções exibe, busca e a leitura já estão prontas (não é para fazer!)

```
int BUSCA (int vetor[], int num) {
        int posicao = -1;
        for (i=0;i<10;i++)
            if (vetor[i] == num)
                posicao = i;
        return posicao;   }
```

Fazer a função DIFERENÇA e o MAIN

```
void DIFERENCA (
    int vetorA[], int vetorB[], int vetorC[])
```

Exemplo: vetorA[] = {36, 2, 4, 8}
vetorB[] = {2, 29, 17,4 }
vetorC[] = {36, 8}



9) Ler 10 elementos do tipo inteiro e armazene-os num vetor de tal capacidade.

a) contar quantos elementos são positivos
b) contar quantos elementos são negativos
c) contar quantos elementos são nulos
d) contar quantos elementos são pares
e) contar quantos elementos são ímpares

10) Ler 10 elementos do tipo inteiro e armazene-os num vetor de tal capacidade.

a) somar todos os elementos armazenados em posições pares
b) somar todos os elementos armazenados em posições ímpares
c) exibir as duas somas calculadas

11)Ler um vetor de inteiros com capacidade para 10 elementos.
Verificar e exibir se o vetor é simétrico. Veja exemplo de um vetor simétrico abaixo:

0	1	2	3	4	5	6	7	8	9
10	20	30	40	50	50	40	30	20	10

Projeto

Crie os seguintes vetores abaixo:

UNIDADE
Um
Dois
Três
Quatro
Cinco
Seis
Sete
Oito
Nove

EXCEÇÃO
Onze
Doze
Treze
Catorze
Quinze
Dezesseis
Dezesete
Dezoito
Dezenove

DEZENA
Dez
Vinte
Trinta
Quarenta
Cinqüenta
Sessenta
Setenta
Oitenta
Noventa

CENTENA
Cem /cento
Duzentos
Trezentos
Quatrocentos
Quinhentos
Seiscentos
Setecentos
Oitocentos
novecentos

O projeto deve utilizar funções com passagem de parâmetros.

Exemplo: Valor numérico..... 37.128.402,14
Valor por extenso.....Trinta e sete milhões, cento e vinte e oito mil, quatrocentos e dois reais e catorze centavos

O menu de opções deve conter:

1 – Ler um valor numérico (real) // valor máximo permitido 99.999.999,99
2 – Transformar para valor por extenso
3 – Sair

Para resolver este projeto, carregue os vetores contendo o numero por "extenso" conforme a posição dígito no valor numérico (centena, dezena, unidade e exceção)

```
static char unidade[10][8]={"zero","um","dois","tres","quatro","cinco","seis","sete","oito","nove"};
static char dezena[10][10]={"zero","dez","vinte","trinta","quarenta","cinquenta","sessenta",
```

```
                                    "setenta","oitenta","noventa"};
  static char centena[10][14]={"zero","cem","duzentos","trezentos","quatrocentos","quinhentos",
                                    "seiscentos","setecentos","oitocentos","novecentos"};
  static char excecao[10][10]={"zero","onze","doze","treze","catorze","quinze",
                                    "dezesseis","dezesete","dezoito","dezenove"};
  char valorT[15];
  printf("\nEntre com um valor monetario <999.999,99>: $");
  scanf("%s",&valorT);
  printf("\n valor lido=%s",valorT);
```

8. MATRIZES

Veja abaixo uma matriz bidimensional representada na memória. Repare nas posições dos elementos. Exemplo de uma matriz 3 x 3 de inteiros.

int mat[3][3];

mat[0][0]	mat[0][1]	mat[0][2]
mat[1][0]	mat[1][1]	mat[1][2]
mat[2][0]	mat[2][1]	mat[2][2]

Ex: Ler uma matriz quadrada 3 x 3. Calcular e exibir a soma da diagonal principal e da diagonal secundária.

```
#include <stdio.h>#include <string.h>
#define tf 3
void leitura(int mat[tf][tf]) {
     int i,j;
     printf("\nLeitura Matriz[3,3] \n");
     for (i=0;i<tf;i++) // linhas I
       for (j=0;j<tf;j++) // colunas J
     {
        printf("\n mat[%d,%d]: ",i,j);
        scanf("%d",&mat[i][j]);
    }
}
```

```
int   diagPrincipal(int  mat[tf][tf])
{
   int soma=0;
   printf("\nDiagonal Principal\nElementos...");
   for (int i=0;i<tf;i++)
    {
        printf("\n Mat[%d,%d]: %d\n",i,i,mat[i][i]);
        soma=soma+mat[i][i];
    }
   return soma;
}
```

```
int   diagSecundaria(int  mat[tf][tf])
{
   int soma=0;
   int j=tf;
   for (int i=0;i<tf;i++)
      {
         j=j-1;
         printf("\n Mat[%d,%d]: %d\n",i,j,mat[i][j]);
         soma=soma+mat[i][j];
      }
   return soma;
}
```

```
main() {
     int matriz[tf][tf];
     int i,j;
     leitura(matriz);
     printf("\n Exibe a matriz lida\n");
     for (i=0;i<tf;i++) {
         for (j=0;j<tf;j++)
               printf("%d  ",matriz[i][j]);
       printf("\n");
     }
     printf("\nResultado Soma Diagonal Principal= %d\n\n",diagPrincipal(matriz));
     printf("\nResultado Soma Diagonal Secundária= %d\n",diagSecundaria(matriz));
      printf("\n\n\n");
      system("PAUSE");
}
```

8.1 EXERCÍCIOS UTILIZANDO MATRIZES

1) Ler uma matriz quadrada 4x4 de inteiros. Descobrir e exibir o maior elemento e a sua posição correspondente.

2) Faça um programa que seja capaz de conhecer e armazenar 10 valores iinteiros quaisquer, em um conjunto bidimensional com dimensões de 2 linhas e 5 colunas.

3) Ler uma matriz quadrada de tamanho 4x4. Calcular a somatória de cada linha da matriz e armazenar o resultado num vetor de 4 posições, conforme ilustração abaixo.

Matriz 4x4

3	5	8	10
9	5	3	1
12	2	4	5
6	20	4	7

Vetor[4]

26
18
23
37

```
#include <stdio.h>#include <string.h>
#define tf 4
void somatoria (int mat[tf][tf], int vetor[])
{
     int i,j;
     for (i=0;i<tf;i++)   {
        soma=0;
        for (j=0;j<tf;j++)  {
          soma=soma+mat[i][j];
       }
        vetor[ i ] = soma;
     }
}
```

4) Faça um programa para leia uma matriz, calcule e exiba outra matriz que seja a transposta da matriz lida.

Obs: é preciso ter o TLL (tamanho lógico das linhas) e o TLC (tamanho lógico das colunas).

Matriz 2x3

A00	A01	A02
A10	A11	A12

Transposta 3x2

A00	A10
A01	A11
A02	A12

5) Faça um programa que efetue a leitura, a soma e a impressão do resultado entre duas matrizes inteiras com 5 linhas e 5 colunas.

6) Escreva um programa que: Leia uma matriz quadrada 10x10 de elementos inteiros, exiba essa matriz e calcule e exiba a soma dos elementos situados abaixo da diagonal principal da matriz, incluindo os elementos da própria diagonal principal.

7) Faça um algoritmo que efetue a troca dos elementos entre as linhas 2 e 4 de uma matriz bidimensional de inteiros (5 x 5) Exemplo:

0	2	3	8	9
7	4	5	6	1
2	3	4	5	6
1	2	1	2	3
5	7	8	9	0

0	2	3	8	9
1	2	1	2	3
2	3	4	5	6
7	4	5	6	1
5	7	8	9	0

8) Faça uma função que seja capaz de conhecer os elementos inteiros de uma matriz, juntamente com os seus tamanhos lógicos e de informar qual linha da matriz tem a maior soma de seus elementos.

9) Veja a matriz abaixo de consumo na lanchonete e seu respectivo vetor de preços associado.

	Rose	Rui	Maria	Vilma	Pedro	Vetor de Preços
refrigerante	0	2	3	8	9	2,00
Coxinha	7	4	5	6	1	1,50
Sorvete	2	3	4	5	6	1,50
Cerveja	1	2	1	2	3	2,80
pinga	5	7	8	9	0	0,50

Faça um programa completo para ler e exibir as informações acima e, além disso, calcular e exibir a conta de cada cliente da lanchonete.

10) Faça um programa em C que implemente o algoritmo conhecido como Quadrado Mágico. O quadrado mágico preenche uma matriz quadrada (5x5) com os números de 1 a 25, seguindo os seguintes passos:

- Zerar a matriz inteira;

- Colocar o Nº 1 na coluna 3, linha 1;
- Fazer COLUNA –1 e LINHA –1 para inserir o Nº 2, e assim sucessivamente:
- Se COLUNA=0 e LINHA >0 então COLUNA=TL
- Se COLUNA>0 e LINHA=0 então LINHA=TL
- Se LINHA=0 e COLUNA=0 ou Matriz[COLUNA,LINHA] <> 0 então LINHA+2 e COLUNA +1

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

11) Considere um vetor de clientes (20 clientes) e outro vetor contendo o preço dos produtos (são apenas 10 produtos).

	Vetor de Clientes
0	Carla
1	Cesar
2	Deise
3	Felipe
4	Gustavo
..	Maria
..	...
...	...
...	...
19	Zuleika

	Preço dos Produtos
0	1,50
1	3,00
2	4,20
3	2,00
4	2,50
5	3,50
6	5,00
7	2,10
8	1,80
9	2,20

E a Matriz de Consumo (inteiros) 20 x 10
Coluna = PRODUTOS. Note que a coluna 0 refere-se ao produto que custa $1,50. Coluna 1 refere-se ao produto que custa $3,00 e assim sucessivamente.
Linha = CLIENTES Note que a linha 0 refere-se ao consumo do Cliente 0 (Carla). Na linha 1 trata-se do consumo realizado pelo Cliente 1 (Cesar) e assim sucessivamente.

Produtos / **Cod.Cliente**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	6	0	0	1	0	0	2	0	0
1	1	3	2		1	0	3	0	0	0
2	4	1	0	0	0	2	4	1	0	0
3	5	0	0	2	0	0	0	0	2	0
4	4	0	1	0	0	0	0	0	0	2
5	9	0	3	1	0	0	2	1	0	1
6	0	2	4	0	0	0	2	1	0	3
7	0	1	0	0	0	0	0	0	2	0
8	2	0	1	0	0	0	3	0	1	1
9	1	0	0	4	0	0	0	2	0	0
10										
....										
19										

Faça um programa completo (utilizando FUNçÃO com passagem de parâmetros para:

a) Faça uma função para calcular a conta de um determinado cliente (de 0 a 9). No programa principal você deve ler o numero do cliente (de 0 a 9) e passar este numero como parâmetro para a função.
totaldocliente = calculaconta (matriz, vetnome, vetpreço, numerolido);

b) Calcule a conta de cada cliente

Cliente 3 Deise

Produto	Quantidade Consumida	Preço Unitário	Valor a Pagar
2	6	$ 3,00	$ 18,00
5	1	$ 2,50	$ 2,50
8	2	$ 2,10	$ 4,20
		Total a Pagar	$ 24,70

c) Calcule e exiba o faturamento do botequim no final do dia (somatório de todas as contas)

----------PROJETO ----------

Dada uma matriz bidimensional (25x25) contendo as distâncias entre as cidades:

	01	02	03	04	05	06	07	08	09	....	25
01	0	2680	1300	3000	....						
02	2680	0	800	500	....						
03	1300	800	0	1800	...						
04	3000	500	1800	0	...						
05	...	...	...		0						
...											
25											

	CIDADES
1	Belo Horizonte
2	Manaus
3	Belém
4	Porto Velho
...	
25	

Exemplo: João deseja viajar de Belo Horizonte para Belém, ou seja,

Belo Horizonte (origem) ---→ código 1 (linha)

Belém (destino) ----------→ código 3 (coluna)

Ao procurar na matriz, na posição linha 1 coluna 3, temos 1300 km de distância.

a) Fazer a declaração das estruturas de dados necessárias para a resolução deste problema.
b) Fazer uma função que receba (via parâmetro) um número de 1 a 25, e retorne o nome da cidade correspondente daquela posição no vetor de cidades
c) Fazer uma função que receba (via parâmetro) dois números (de 1 a 25). O primeiro número corresponde ao número da cidade origem, e o segundo número corresponde ao

número da cidade destino. Descubra e retorne como resultado da função a distância entre as duas cidades (localize na matriz). Também imprima (dentro da função) os nomes das cidades origem e destino e a distância entre elas.

d) Fazer uma função que exiba todas as distâncias inferiores a 500 km. O formato da saída deve ser: Cidade Origem – Cidade Destino – Distância.
e) Dado o percurso entre as cidades A,B,C, D,E,F,G exiba o total percorrido e o nome das cidades visitadas.
f) Exiba a matriz de distâncias sem repetições, isto é, se a distância entre as cidades A e B já foi exibida, não é necessário exibir a distância entre B e A.

9. ESTRUTURAS OU REGISTRO (STRUCT)

Ao manusearmos dados muitas vezes deparamos com informações que não são fáceis de armazenar em variáveis escalares como são os tipos inteiros e pontos flutuantes, mas na verdade são conjuntos de coisas. Este tipo de dados são compostos com vários dos tipos básicos do C. As estruturas permitem uma organização dos dados dividida em campos e registros. Ex:

1) Criar uma estrutura em C que armazene informações sobre um produto

```
struct produto {
        char nome[30];
        char descricao[30];
        float preco;
        int quant;
    };
struct produto mercearia[15]; // vetor de 15 produtos chamado mercearia
```

2) Definir a estrutura cuja representação gráfica é dada a seguir, definir os campos com os tipos básicos necessários (STRUCT)

Cadastro

Nome	Endereço	Salário	Idade	Sexo	Telefone

Endereço

Rua	Bairro	Cidade	Estado	cep

3) (exemplo resolvido) As contas dos clientes de um banco podem ser divididas em contas “Comum” e “Especial”. Clientes com contas especiais possuem um campo para o valor do limite, ao passo que contas do tipo Comum não. Além disso, todas as contas devem conter: numero, nome do cliente e saldo.

```
#include <string.h>
struct conta{
    int num_conta;
    char tipo_conta;
    char nome[80];
    float limite;
    float saldo
};
```

```
main() {
        int i;
        struct conta cliente[3];
          cliente[0].num_conta=110;
          cliente [0].tipo_conta='C';  // conta comum
          strcpy(cliente [0].nome,"Joao Henrique") ;
          cliente [0].limite = 0;
          cliente [0].saldo = 1500.43;

          cliente[1].num_conta=201;
          cliente [1].tipo_conta='E';  // conta especial
          strcpy(cliente [1].nome, "Carlota Joaquina") ;
          cliente [1].limite = 1000.00;
          cliente [1].saldo = 8500.75;

          cliente[2].num_conta=310;
          cliente [2].tipo_conta='C';  // conta comum
          strcpy(cliente [2].nome, "Marcelo Pereira") ;
          cliente [2].limite = 0;
          cliente [2].saldo = 780.90;

        printf("\n\n Nro.Conta  Tipo  Cliente  Saldo\n");
        for(i=0;i<3;i+ +){
            if (cliente[i].tipo_conta=='C')
              printf("%d\t%c\t\t%s\t%f\n", cliente[i].num_conta, 'Conta Comum',
                                              cliente[i].nome,cliente[i].saldo);
             else
              printf("%d\t%c\t\t%s\t5f\t%f\n",cliente[i].num_conta,'Especial' ,
                                             cliente[i].nome,cliente[i].limite, cliente[i].saldo);
        }//for
}//main
```

4) (exercício) Suponha a seguinte estrutura referente a um aluno:

```
struct aluno {
        int matricula;
        char nome[20];
        float nota1, nota2, nota3;
        int frequencia;
};
```

Supor a existência de 10 alunos. MediadoAluno = ∑ (nota1 + nota2 + nota3) /3
Aprovado se (média >= 6) e (freqüência >=75), caso contrário Reprovado.
Faça um programa completo que utilize funções com passagem de parâmetros para cada item do menu solicitado (MENU PRINCIPAL) contendo:

1 – Carrega o vetor de alunos
2 – Exibir o vetor de alunos carregado
3 – Ler uma matricula e fazer BUSCA no vetor para exibi-lo
4 – Exibir todos os alunos aprovados
5 – Exibir todos os alunos reprovados
6 – Calcular e exibir a média geral de todos os alunos
7 - Sair

5) Refaça o exercício 3 (clientes de banco) com os seguintes acréscimos:

a) crie um vetor de 10 posições para armazenar os registros dos clientes

b) faça um menu no programa principal (main) contendo as seguintes opções (cada opção deverá chamar uma função correspondente no programa, utilizando passagem de parâmetros):

```
printf ("\n\n\n1- Inserir uma conta de cliente");
printf ("\n2- exibir todas as contas");
printf ("\n3- Procurar uma conta e exibi-la");
printf ("\n4- Deposito de uma determinada conta");
printf ("\n5- Retirada/saque em uma determinada conta");
printf ("\n6- Remove uma conta");
printf ("\n7- Sair");
```

----------RESOLUÇÃO ----------

```
struct conta{
    int num_conta;
    char tipo_conta;
    char nome[30];
    float limite;
    float saldo;
};
int busca (conta clientes[10],int tam,int num)
{
   int i=0;
   while ((i<tam)&&(num!= clientes[i].num_conta))
      i++;
   if ((i<tam)&&(num == clientes[i].num_conta))
      return i;
      else
      return -1;
}

void carrega(conta cliente[10]){
  cliente[0].num_conta=110;
  cliente [0].tipo_conta='C';  // conta comum
  strcpy(cliente [0].nome,"Joao Henrique") ;
  cliente [0].limite = 0;
  cliente [0].saldo = 1500.43;

  cliente[1].num_conta=201;
  cliente [1].tipo_conta='E';  // conta especial
  strcpy(cliente [1].nome, "Carlota Joaquina") ;
  cliente [1].limite = 1000.00;
  cliente [1].saldo = 8500.75;

  cliente[2].num_conta=310;
  cliente [2].tipo_conta='C';  // conta comum
  strcpy(cliente [2].nome, "Marcelo Pereira") ;
  cliente [2].limite = 0;
  cliente [2].saldo = 780.90;
}
```

```
void insere(conta cliente[10], int tl){
   printf ("entre com o numero da conta");
   scanf("%d",&cliente[tl].num_conta);
   printf ("entre com o tipo da conta (C/E)");
   cliente[tl].tipo_conta = getchar();
   printf ("entre com o nome do cliente");
   scanf("%s",&cliente[tl].nome);
   if (cliente[tl].tipo_conta=='E'){
     printf ("entre com o limite");
     scanf("%f",&cliente[tl].limite);
     } else
        cliente[tl].limite =0;
   printf ("entre com o saldo");
   scanf("%f",&cliente[tl].saldo);
}

void exibe(conta cliente[10], int tl){
  printf("\n\nConta    Tipo              Cliente              Limite   Saldo\n");
  int i;
  for(i=0;i<tl;i++){
     if (cliente[i].tipo_conta=='C')
          printf("%d\t%s\t\t%s\t\t\t%8.2f\n",cliente[i].num_conta, "Conta Comum",
                        cliente[i].nome,cliente[i].saldo);
          else
          printf("%d\t %s\t\t%s\t%7.2f\t %7.2f\n",cliente[i].num_conta,"Especial" ,
                        cliente[i].nome,cliente[i].limite, cliente[i].saldo);
  }//for
}
main()  {
 int i;
 static struct conta cliente[10];
 carrega(cliente);
 int tl=3;
 int opcao=0, num, posicao;
 float saque, valor;

 while (opcao != 7){
   printf ("\n\n\n1- Inserir uma conta de cliente");
   printf ("\n2- exibir todas as contas");
   printf ("\n3- Procurar uma conta e exibi-la");
   printf ("\n4- Deposito de uma determinada conta");
   printf ("\n5- Retirada em uma determinada conta");
   printf ("\n6- Remove uma conta");
   printf ("\n7- Sair");
   printf ("\nOpcao: ");     scanf("%d",&opcao);
   switch (opcao){
      case 1:  insere(cliente,tl);
               tl++;
               break;
      case 2:  exibe(cliente, tl);     break;
      case 3: // busca uma determinada conta
```

```
    printf ("entre com o numero da conta");scanf("%d",&num);
    posicao = busca(cliente,tl,num);
    if (posicao==-1)
      printf("Conta NAO existe");
      else   printf ("%d\t %c\t\t%s\t%7.2f\t %7.2f\n",cliente[posicao].num_conta,
                    cliente[posicao].tipo_conta,
                    cliente[posicao].nome,cliente[posicao].limite,
                    cliente[posicao].saldo);                break;
case 4://deposito
    printf ("entre com o numero da conta");scanf("%d",&num);
    posicao = busca(cliente,tl,num);
    if (posicao==-1)
      printf("Conta NAO existe");
      else
      {
        printf("Conta do sr(a).%s\n",cliente[posicao].nome);
        printf("entre com o valor do deposito");scanf("%f",&valor);
        cliente[posicao].saldo += valor;
      }
    break;
 case 5://saque
    printf ("entre com o numero da conta");scanf("%d",&num);
    posicao = busca(cliente,tl,num);
    if (posicao==-1)
      printf("Conta NAO existe");
      else
      {
        printf("Conta do sr(a).%s\n",cliente[posicao].nome);
        printf("entre com o valor do saque:");scanf("%f",&saque);
        float saldototal=0;
        if (cliente[posicao].tipo_conta=='E')
            saldototal = cliente[posicao].saldo + cliente[posicao].limite;
            else saldototal = cliente[posicao].saldo;
        if (saldototal >= saque) {
            cliente[posicao].saldo -= saque;
            printf("\n Saque realizado!Novo saldo: $%5.2f",cliente[posicao].saldo);
            } else printf("\nSaldo insuficente. $%5.2f",cliente[posicao].saldo);
      }//else
    break;
 case 6://remove uma conta
    printf ("entre com o numero da conta a ser removida");
    scanf("%d",&num);
    posicao = busca(cliente,tl,num);
    if (posicao==-1)
      printf("Conta NAO existe");
      else {
        for (i=posicao+1;i<tl;i++)
           cliente[i-1]=cliente[i];
        tl--;
        printf("\nRemovido com sucesso");
          }
```

```
            break;
    }//switch
} //while
  system ("PAUSE");
}//main
```

6) Escreva um código em C para fazer a criação dos novos tipos de dados conforme solicitado a seguir:
   Horário: composto por Hora, Minutos e Segundos
   DATA: composto por dia, mês e ano
   Compromisso: composto por uma data, horário e texto que descreve o compromisso

7) Faça um programa que armazene um registro de dados (estrutura composta) os dados de um funcionário de uma empresa, contendo: Nome, idade, sexo (M/F), CPF, data de nascimento, código do setor aonde trabalha (0-99), cargo que ocupa (string de até 30 caracteres) e salário. Os dados devem ser digitados pelo usuário, armazenados na estrutura e exibidos na tela.

8) Faça um programa que leia os dados de 10 alunos (nome, matricula, média final), armazenados num vetor. Uma vez lido os dados, divida estes dados em 2 novos vetores, o vetor dos aprovados e o vetor dos reprovados. Considere a média mínima para a aprovação como sendo 6.0. Exibir na tela os dados do vetor de aprovados, seguido dos dados do vetor de reprovados.

9) Faça um programa que gerencie o estoque de um mercado e:
   - Crie e leia um vetor de 5 produtos, com os dados: código (inteiro), nome (Max de 15 letras), preço e quantidade;
   - Leia um pedido, composto pelo código de produto e quantidade. Localize (busca) este código no vetor e, se houver quantidade suficiente para atender ao pedido integralmente, atualize o estoque e informe o usuário. Repite este processo até ler um código igual a zero.

   OBS: fazer um menu de opções no programa principal (MAIN). Cada item do menu deve corresponder a uma função implementada utilizando parâmetros. Faça a função de busca que retorna a posição do código procurado dentro do vetor.

10) Faça um programa que controle o fluxo de vôos de aeroportos de um pais. Suponha 5 voos e 5 aeroportos. Faça:
   - Crie e leia um vetor de vôos, sendo que cada vôo contem um código de aeroporto de origem e um de destino;
   - Crie um vetor de aeroportos, sendo que cada aeroporto contem seu código, quantidade de vôos que saem e quantidade vôos que chegam.

   NOTA: cada aeroporto é identificado por 3 letras (BRA=Brasília; PVH=porto velho; CGH = congonhas; REC=recife;GRU=Guarulhos)

11) Criar 3 estruturas de dados para atender o seguinte:

a) A primeira estrutura deve conter o nome de uma pessoa com a respectiva data de nascimento (ver struct´s data e pessoa abaixo e também a Matriz 20x2)

b) A segunda estrutura contêm o nome do signo com as datas (dia e mês) de inicio e fim e descrição das características correspondentes ao signo (ver matriz do Horoscopo Ocidental abaixo).

c) A terceira estrutura que contem o nome do signo no horóscopo chinês, um vetor contendo 6 anos e a descrição das características correspondente ao signo (Apesar do horóscopo chinês considerar também dia e mês de nascimento, para este exemplo, somente nos preocuparemos com o ano).

OBS: para esse exercício será necessário criar struct e também, vetores contendo struct

Matriz com o nome e a data de nascimento (20x2) – item a

	Nome	Data nascimento (dia/mês/ano)
0	Maria	dia mes ano
1	João	dia mes ano
...	.....	
19	Vilma	

```
struct data{
    int dia;
    int mes;
    int ano;
 };

struct pessoa{
    char nome[10];
    struct data nascimento;
 };
```

Matriz do Horóscopo Ocidental (12x6) – item b

	Signo	Dia Inicio	Dia Final	Mês Inicio	Mês Final	Característica
0	Aries	21	20	3	4	Ação, impetuosidade,
1	Touro	21	20	4	5	Calma, possessividade
2	Gemeos	21	20	5	6	Comunicativo
...3	Câncer	21	20	6	7	Instável
4	Leão	21	20	7	8	Autoritário
5	Virgem	21	20	8	9	Nervoso
6	Libra	21	20	9	10	Equilibrado
7	Escorpião	21	20	10	11	Vingativo
8	Sagitário	21	21	11	12	Viajante
9	Capricornio	22	20	12	01	Persistente
10	Aquario	21	19	01	02	Racional
11	Peixes	20	20	02	03	Intuitivo, sentimental

```
struct horoscopo {
        char signo[10];
        int diaInicio;
        int diaFinal;
        int mesInicio;
        int mesFinal;
        char caracteristica[50];
};
```

Matriz do Horoscopo Chinês (12x3) – item c

	Signo	Anos (vetor de 6 posições)	Descrição
0	Rato	1948 | 1960 | 1972 | 1984 | 1996 | 2008	Criativo, trabalhador
1	boi	1948 | 1960 | 1972 | 1984 | 1996 | 2008	Gentil, pacífico
...	....		
11			

```
struct chines {
        char signo[10];
        int anos[6];
        char descricao[50];
};
```

PROJETO

vetPRODUTOS

	Descrição	Codigo do Fornecedor	Qtdade Estoque	Qtdade Minima	Qtdade Maxima
0	Arroz	777	100	150	200
1	Feijão	777	190	60	100
2	Leite condensado	444	45	50	90
3	Macarrão	555	68	60	100
4	Molho tomate	222	24	20	70
5	Açúcar	333	67	70	100
6	Bolacha salgada	555	135	45	90
7	Água mineral 1 litro	987	60	40	80
8	Farinha de trigo	111	230	40	90
9	Leite CX– 12 unidades	666	20	30	50

Declare a estrutura (STRUCT) para o vetor acima ilustrado.
Declare a variável (VetPRODUTOS), contendo vários produtos (ver tabela).
O vetProdutos tem tamanho fixo (TF) de 50 posições, e uma variável TL para controlar as posições ocupadas. **DECLARE AS ESTRUTURAS envolvidas**.

a) **Faça uma função para exibir todos os produtos que estão abaixo do estoque**. Para cada produto exibido (cuja quantidade no estoque está abaixo do mínimo) você deve calcular qual a quantidade a ser solicitada para o Fornecedor.

IF (QtdadeEstoque < QtdadeMinima) fazer pedido ao fornecedor
qtdadeSolicitada = QtdadeMaxima – QtdadeEstoque

Arroz fornecedor:777 QtdMax=200 QtdEst=100 pedir a quantidade: 100
Leite condensado fornecedor:444 QtdMax=90 QtdEst=45 pedir a quantidade: 45
Açúcar fornecedor:333 QtdMax=100 QtdEst=67 pedir a quantidade: 33

b) **Faça uma função para exibir todos os produtos que estão EXCEDENTES no estoque** (a quantidade no estoque está acima do máximo permitido). Exemplo:
feijão fornecedor: 777 estoque atual: 190 estoque máximo permitido: 100 excedente: 90
bolacha salgada fornecedor: 555 estoque atual: 135 estoque máximo permitido: 90 excedente: 45
Farinha de trigo fornecedor: 111 estoque atual: 230 estoque máximo permitido: 90 excedente: 140

c) **Faça uma função para REMOVER um produto do vetPRODUTOS**. Deve ser lido o código do produto no programa principal (MAIN) e passado como parâmetro para a função REMOVER.

Você deve implementar e utilizar a seguinte função
int posição = BUSCA (Produto vetProdutos[], int TL, int codigoprocurado)
Repare que esta função faz uma busca dentro do vetPRODUTOS.
A função busca o "codigoprocurado" e retorna a posição (linha) aonde encontrou (se não encontrar, retorna -1).

10. ARQUIVOS DE DADOS

No capítulo 9 estudamos estruturas de dados heterogêneas (STRUCT). Nos exemplos estudados, o armazenamento das informações foi realizado na memória principal. A desvantagem na utilização da memória principal é que, ao encerrarmos o programa, todas as informações digitadas são automaticamente perdidas.
Agora, estudaremos o armazenamento de dados em memória secundária utilizando arquivo de dados. Neste capítulo, abordaremos o tipo TEXTO (vamos utilizar o bloco de notas para criá-lo).

Comandos para abertura de arquivos

a) Comando necessário que permite que as informações sejam armazenadas temporariamente para realizar a transferência entre a memória do computador e o arquivo de dados.

FILE *arq;

b) Comando de abertura do arquivo (leitura e/ou gravação de dados)

arq = fopen ("*nomeArquivo.txt*", "*tipo*");

Tipo	Significado
“r”	Abre um arquivo (texto) para leitura. O arquivo deve existir antes de ser aberto.
“w”	Abre um novo arquivo texto para gravação. Se o arquivo já existir, ele será destruído e o novo arquivo será criado no seu lugar.
“a”	Abre um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo (append = acréscimo de novos registros no final). Um novo arquivo será criado, caso o arquivo não exista
“r+”	Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado.
“w+”	Cria um arquivo texto para leitura e para gravação. Se o arquivo já existir, ele será destruído e um novo arquivo será criado no seu lugar.
“a+”	Abre um arquivo texto para leitura e gravação. Os dados serão adicionados no final do arquivo se ele já existir ou um novo arquivo será criado, no caso dele não existir.

Exemplo 1:

```
FILE *fp; /* declaração de um arquivo
fp= fopen(“listatelefones.txt”,”r”); /* o arquivo se chama listatelefones.txt para leitura. O arquivo deve existir antes de ser aberto */
if (!fp) {
    printf(“\n\nErro na abertura do arquivo”);
    exit(0);  getchar();
}
```

As funções ***fprintf*** e ***fscanf*** são utilizadas para escrever e para ler no arquivo.
A função ***fclose()*** é utilizada para fechar o arquivo.
Função *feof()* (função EOF End Of File) indica o fim do arquivo. As vezes é necessário verificar se um arquivo chegou ao fim. Para isso, podemos usar a função feof(arq). Ela retorna não-zero se arquivo chegou ao EOF, caso contrário retorna zero.

Exemplo 2: Gravar os números de 1 até 10 no arquivo teste.txt

```
#include <stdio.h>
main() {
  FILE *f = fopen("teste.txt", "w");
  int i;
  for (i=1; i<=10; i++)
        fprintf(f, "%d\n", i);
  fclose(f);
}
```

OBSERVAÇÃO: o arquivo texto “teste.txt” fica na mesma pasta aonde se encontram o programa fonte e o programa compilado.

Exemplo 3: Ler e exibir os números que estão no arquivo teste.txt

```
#include <stdio.h>
main( ) {
    FILE *f = fopen("teste.txt", "r");
     int i;
     while (fscanf(f, "%d", &i) == 1) // se igual a 1 tem-se EOF (fim do arquivo)
              printf("%d\n", i);
     fclose(f);
}
```

Exemplo 4: Em um arquivo chamado "notas.txt" estão os dados nomes e notas de alunos.
Em cada linha há o nome do aluno, seguido de três notas:

Maria 8 8 10
Jose 6 6 8
Camila 7 9.5 7.5
Programador 10 10 10

Crie um programa que exiba o nome de cada aluno e sua média.Observe que em toda linha, há um padrão:

string, espaço, número, espaço, número, espaço, número, enter

Ou seja: **"%s %f %f %f\n"**

```
#include <stdio.h>
main() {
        char  nome[20];
        float  nota1, nota2, nota3, media;
        FILE *arq = fopen("notas.txt", "r");
        if(!arq)
                {   printf("Erro, nao encontrou o arquivo\n");   exit(0);   getchar();   }
        else  {
                while( (!feof(arq) ){
                        fscanf(arq,"%s %f %f %f\n", nome, &nota1, &nota2, &nota3);
                        media = (nota1 + nota2 + nota3) /3;
                         printf("%s teve media %.2f\n", nome, media);
                    }
                fclose(arq);
            }//else do if
}
```

REFAÇA o EXERCICIO ACIMA UTILIZANDO MENU

1 – LEITURA DO ARQ TEXTO E CARREGA VETOR
2 – EXIBE O VETOR CARREGADO
3 – EXIBE TODOS OS APROVADOS
4 – INSERE NOVO ALUNO NO FINAL DO VETOR
5 – GRAVA O VETOR NO ARQUIVO TEXTO
6 – SAIR

EXERCÍCIO para laboratorio:

PASSAGEIRO 1 – 10	EMBARQUE 12-13	DESTINO 15-16
Carmem	SP	RJ
Lucila	MT	BA
Vera	RJ	RS
Luís Carlos	BA	PR
Benjamin	AM	PA

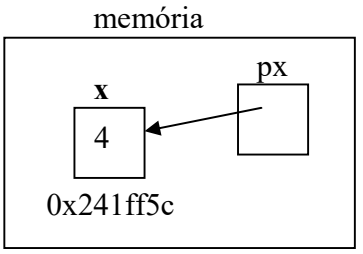
Trabalhar com arranjos complementares (pares).
Primeiramente ler as informações de um arquivo texto e armazená-las em arranjos complementares.
Fazer um procedimento para ler o nome de um passageiro X. Informar se ele está entre os passageiros do vôo, onde ele irá embarcar e qual o seu destino.

Menu:
1- Le arquivo texto e armazena em arranjos
1- Exibe os arranjos
2- Consulta de embarque - método LOCALIZA
3- Sair

11. PONTEIROS

Ponteiro é uma variável que contém o *endereço* de um objeto de dados, em geral uma outra variável. Essa é a razão para o seu nome: ele *aponta* para outra variável.

```
// exemplo de utilização de PONTEIROS   #include <stdio.h>
        int main() {
            int x = 4;
            int *px ;
            px = &x;
            printf( "\n Endereco  px = &x = %X", px);
            printf("\n Conteudo *px =  x = %d",*px);
            system("PAUSE");
            return 0;
        }
```




```
Endereco  px = &x = 22FF74
Conteudo *px =  x = 4
```

A variável **x** é do tipo inteiro. O conteúdo da variável **x** é igual a 4.

A variável **x** ocupa um endereço na memória do computador (por ex. endereço 0x241ff5c).

Para obter o endereço da variável **x** dentro da memória, utilizamos **&x** (&x = 0x241ff5c).

A variável **px** é um ponteiro. Repare na declaração do ponteiro (int ***px**).

A variável **px** recebe o endereço da variável **x** (px = &x).

Repare que o conteúdo de **px** não é 4, mas sim o endereço da variável **x** (px = 0x241ff5c).

Dizemos que **px** aponta para **x**.

px representa o endereço de **x**, não o seu valor (px = 0x241ff5c).

***px** significa o conteúdo da variável para o qual px aponta (no caso, *px = 4).

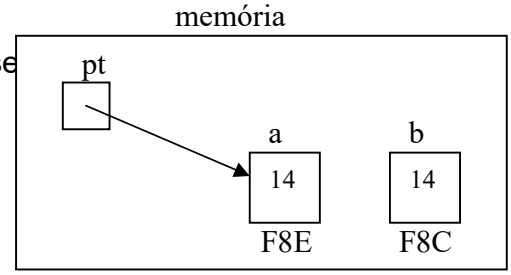
Ao executarmos o programa acima, teremos o seguinte dados exibidos na tela:

```
Endereço   px = &x = 0x241ff5c
Conteúdo *px =   x =  4

Pressione qq tecla para continuar...
```

Exercícios de Ponteiros

1) Faça um teste de mesa com o exemplo a se




```
void main() {
    int a, b;
    int *pt=&a;
    *pt=14;   // equivale comando a=14
    b = *pt;   // equivale comando b = a
    printf("\n a = %d",a);   /* exibe a= 14 */
    printf("\n &a = %p",&a); /* exibe endereço hexadecimal &a=F8E */
    printf("\n pt = %p", pt);  /* exibe endereço hexadecimal pt=F8E */
    printf("\n *pt = %d", *pt); /* exibe *pt=14 */
    printf("\n b = %d",b);
    system("PAUSE");
}
```

2) Faça o teste de mesa

```
- void main() {
-     int num, valor;
-     int *pt;
-     num =55;
-     pt = &num; /*  pega o endereço de num */
-     valor = *pt;   /* valor é igualado a num de maneira indireta */
-     printf("\n %d \n", valor);
-     printf("\n Endereço para onde o ponteiro aponta %p \n", pt); /*exibe endereço hexadecimal*/
-     printf("\n Valor da variável apontada %d\n", *pt); /* exibe 55 */
-     system("PAUSE");
- }
```

3) Faça o teste de mesa:

```
- void main() {
-     int num, *pt;
-     num =55;
-     pt = &num; /*  pega o endereço de num */
-
-     printf("\n valor inicial: %d \n", num);
-     *p=100; /*muda o valor de num de uma maneira indireta */
-     printf("\n Valor final %d\n", num); /* exibe 100 */
-     system("PAUSE");
- }
```

// **exemplo de utilização de PONTEIROS**

```
#include <string.h>
#include <stdio.h>
    struct no{
        char info[10];
        no *prox;
    };
main(){
  no *atual = (no*)malloc(sizeof(no));
  no *novo = (no*)malloc(sizeof(no));
```

```
Atual: Maria
novo: Joao
pont:Pedro

Atual:Maria
Prox do atual:Joao
Prox do Prox do atual:Pedro

Dando a volta circular...Prox do pont:Maria_
```

```
    strcpy(atual->info,"Maria");
    strcpy(novo->info,"Joao");
    atual->prox=novo;

    no *pont = (no*)malloc(sizeof(no));
    strcpy(pont->info,"Pedro");
    novo->prox = pont;

    printf("\n  Atual: %s",atual->info);
    printf("\n  novo: %s",novo->info);
    printf("\n  pont:%s",pont->info);
    getchar();

    printf("\n  Atual:%s ",atual->info);
    printf("\n  Prox do atual:%s ",atual->prox->info);
    printf("\n  Prox do Prox do atual:%s ",atual->prox->prox->info);
    getchar();

    pont->prox = atual;
    printf("\n  Dando a volta circular...Prox do pont:%s",pont->prox->info);
    getchar();
}
```