- Iniciaremos o uso do Dev C++ C compilando e executando um programa simples.
- Considere o algoritmo abaixo e seu equivalente em C:

```
Algoritmo
var a,b,soma : inteiro
inicio
   escreva("Digite um valor")
   leia(a)
   escreva("Digite outro valor")
   leia(b)
   soma<-a+b
   escreva("Soma=",soma)
fimalgoritmo
```

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{
   int a,b,soma; /* declaração de variáveis */
   printf("Digite um valor:");
   scanf("%d",&a);
   printf("Digite outro valor:");
   scanf("%d",&b);
   soma=a+b;
   printf("Soma=%d\n",soma);
   system("pause");
}
```

Abra o Dev C++ e execute os passos a seguir:

- 1) Abra uma janela de edição (Arquivo $\rightarrow$ Novo $\rightarrow$ Arquivo Fonte)
- 2) Digite o programa acima (ctrl-c ctrl-v também funciona)
- 3) Salve o programa digitado (Arquivo $\rightarrow$ Salvar como)
- 4) Compile o programa digitado (Executar $\rightarrow$ Compilar)
- 5) Se não ocorreram erros de sintaxe, execute o programa (Executar $\rightarrow$ Executar)

Observações:

- 1) A linguagem C diferencia entre minúsculas e maiúsculas.
- 2) O erro de sintaxe mais comum na linguagem C é a falta de ponto-e-vírgula. O ponto e vírgula é obrigatório ao final de cada comando.

E agora, uma rápida explicação de cada um dos elementos do programa:
1) Inclusão de bibliotecas

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

- A linguagem C possui várias bibliotecas (conjuntos de funções prontas).
- Bibliotecas para leitura e escrita, matemáticas, para manipulação de textos, etc.
- Para usar funções de uma biblioteca, é necessário informar no início do programa quais bibliotecas serão utilizadas.
- Isso é feito com a cláusula #include <nome da biblioteca>.
- No nosso programa, a biblioteca stdio.h possue as funções de leitura e escrita (printf e scanf), e a biblioteca stdlib.h possui a função system, utilizada no final do programa para que ele espere o usuário digitar algo.

2) Bloco do programa principal (Função main)

```
int main( )
{

}
```

Um programa em C é formado por um conjunto de blocos (chamados funções). Um dos blocos é chamado de main e é por onde inicia a execução do programa. O formato de cada bloco é

```
tipo_de_valor_retornado_pelo_bloco  nome_da_função  (parâmetros)
{
}
```

3) Declarações de variáveis

- As declarações de variáveis em C tem o formato:
- tipo lista_de_variáveis;
- E os principais tipos são int (variáveis inteiras) e float (variáveis reais). Ex:
- int c1,b1,cont;
- float peso, media;
- Da mesma forma que em algoritmos, nomes de variáveis devem iniciar por letra e podem conter letras, dígitos e o caracter de sublinha (_).

4) Comentários

- Comentários podem ser escritos no código para melhorar sua clareza. Comentários são formados por qualquer sequência de caracteres delimitados por /* ..... */
- Os compiladores da linguagem C normalmente aceitam (apesar de não ser parte do C padrão)também comentários de linha, que iniciam por // e se estendem até o fim da linha
- Ex:

```
int a,b,soma; /* Sou um comentário!!! */
printf("Digite um valor:"); // Eu também!!!
```

5) Comando de escrita: printf( )

- O comando correspondente ao comando "escreva" é a função printf(), presente na biblioteca stdio.h (não esqueça o #include <stdio.h>
- A função printf( ) tem como primeiro parâmetro a mensagem a ser mostrada na tela.
- Ex: printf("Oi").
- Se essa mensagem deve conter o valor de variáveis ou o resultado de expressões, as variáveis e expressões são listadas imediatamente após a mensagem. Ex:

```
a=3;
b=5;
printf("A soma de %d e %d vale %d",a,b,a+b).
```

- O exemplo anterior mostrará a mensagem "A soma de 3 e 5 vale 8".
- Para exibir valores de variáveis e expressões, a mensagem deve conter uma indicação do local na mensagem onde serão inseridos os valores das variáveis e expressões.
- Esses indicadores de posição tem o formato %d, se o valor a ser inserido for inteiro, e %f se o valor a ser inserido for float.
- Os indicadores de posição devem ser em mesmo número dos elementos a serem inseridos, e na mesma ordem.
- Também podem ser utilizados caracteres de formatação na mensagem. O principal caracter é o $\backslash n$ que causa uma quebra de linha no ponto onde foi inserido.

- Códigos para formatação de impressão das expressões que devem ser passadas para o especificador de formato:

  %c - caractere simples
  %d ou %i - inteiro decimal com sinal
  %e ou %E - notação científica
  %f - ponto flutuante
  %g ou %G - %e ou %f (o mais curto!)
  %o - octal
  %s - cadeia de caracteres
  %u - inteiro decimal sem sinal
  %x ou %X - hexadecimal
  %p - ponteiro (endereço)
  %n - ponteiro (inteiro)
  %% - escreve o símbolo %

5) Comando de leitura: scanf( )

- O comando equivalente ao comando leia( ) é o comando scanf( ).
- Os parâmetros do scanf são um especificador dos tipos dos valores que serão lidos, seguido da lista de variáveis que receberão os valores, sendo que o identificador de cada variável deve ser acompanhado de &.
- Ex:

```
scanf("%d%f",&a,&p);
```

- O exemplo acima recebe 2 valores do teclado, sendo o primeiro um valor inteiro e o segundo um valor float.
- Os dois valores são armazenados respectivamente nas variáveis a e p.

Especificadores de formato

Especificador	Descrição	Caracteres lidos
i	inteiro	Qualquer quantidade de dígitos, opcionalmente precedidos por + ou -. Por padrão os dígitos são decimais (0 a 9), mas um prefixo 0 indica que são caracteres octais (0 a 7) e um prefixo 0x indica que são caracteres hexadecimais (0 a F).
d ou u	inteiro decimal	Qualquer quantidade de dígitos decimais (0-9), opcionalmente precedidos por sinal (+ ou -). d é para argumento com sinal, u para sem sinal
o	inteiro octal	Qualquer quantidade de dígitos octais (0-7), opcionalmente precedidos por sinal (+ ou -).
x	int hexadecimal	Qualquer quantidade de dígitos hexadecimais (0-9,a-f,A-F), opcionalmente precedidos por 0x ou 0X, e opcionalmente precedidos por sinal (+ ou -).
f, e, g,a	float	Uma série de dígitos decimais, opcionalmente contendo um ponto decimal, opcionalmente precedido de sinal (+ ou -) e opcionalmente seguido por e ou E e um número decimal. Algumas implementações suportam floats em hexadecimal, quando precedidos por 0x ou 0X.

Especificadores de formato (continuação)

Especificador	**Descrição**	**Caracteres lidos**
c	caracter	Se um comprimento maior que 1 é especificado, a função le a quantidade de caracteres especificada e armazena nas primeiras posições do vetor passado como parâmetro, sem acrescenter o \0 ao final.
s	string de caracteres	Qualquer quantidade de caracteres exceto o espaço em branco. Um \0 é automaticamente adicionado ao final da sequencia.
p	Pointer	Uma sequencia de caracteres representando um endereço de memória. O formato específico depende do sistem usado, mas é o mesmo usado no %p no printf.
[caracteres]	sequencia de caracteres	Qualquer número de caracteres dentre os especificados entre os colchetes
[^ caracteres]	sequencia de caracteres	Qualquer número de caracteres quaisquer, exceto os especificados entre os colchetes
% %	%	Dois % seguidos representam um %

6) Comando de atribuição:

- Tem o formato:

  identificador_da_variável = expressão;

- Os principais operadores aritméticos da linguagem C são o de adição (+), subtração (-), multiplicação (*), divisão (/).
- É importante, ao escrever uma expressão, considerar os tipo de operandos e o tipo do resultado da operação.
- Os 4 operadores resultam int se ambos os operandos são int, caso contrário, resultam float.

- É importante também considerar a precedência (prioridade) dos operadores.
- Os operadores *, / e % tem a mesma prioridade entre si, que é maior que a prioridade dos operadores + e - (subtração).
- O operador "-" também é utilizado como operador de troca de sinal, e nesse caso tem uma prioridade maior que todos os outros.
- A tabela a seguir resume os principais operadores aritméticos em C:

Operação	Operador	Prioridade
Troca de Sinal	-	1
Multiplicação	*	2
Divisão	/	2
Resto da divisão inteira	%	2
Soma	+	3
Subtração	-	3

- Observações:
  - O operador de troca de sinal é o único operador de apenas um operando
  - O operador de resto de divisão inteira só é aplicável a operandos inteiros

- Parênteses são utilizados para mudar a ordem de avaliação dos operadores em uma expressão.
- A linguagem C faz conversão automática de tipos.
- Pode-se atribuir um int para uma variável float e vice-versa. Ao atribuir um float para um int, as casas decimais são truncadas.

7) O system("pause");

- É utilizado no Dev C++ para que o programa fique esperando que o operador tecle enter.
- A inexistência dessa chamada faria com que o programa terminasse e a janela onde ele é executado fosse fechada antes que o usuário visse o resultado da sua execução.
- A função system está na biblioteca stdlib.h.

Faça os programas a seguir:

1. Faça um programa que leia 3 valores reais, notas de um aluno, e escreva sua média aritmética. A média aritmética de um conjunto de valores é dada pela soma dos valores dividido pela quantidade de valores considerados.
2. Faça um programa que leia 2 valores reais v1 e v2 e calcule e escreva a área do triângulo que tem base igual a v1 e altura igual a v2. Dica:A área de um triângulo é dada pela expressão: (base x altura)/2

Principais funções da Biblioteca Math (#include <math.h>)

- Todas elas retornam valores tipo float.
  - pow (x,y) – Calcula x elevado a y.
  - sqrt(x) – Calcula a raiz quadrada de x.
  - fabs (x) – Calcula o valor absoluto de x.
- Ex:

  a = pow(3,2);
  a = sqrt(b);

3) Faça um programa que leia 3 valores a, b e c, coeficientes de uma equação de segundo grau, e calcule e escreva as raizes da equação. As raízes de uma equação podem ser calculadas pela fórmula de Baskhara:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- Teste seu programa com os coeficientes

  a=2

  b=5

  c=2

- que deve resultar em raízes x_1=-2.0 e x_2=-0.5

4) Faça um programa que leia 2 valores e escreva o maior deles. O maior entre dois valores quaisquer v1 e v2 pode ser calculado pela expressão

$$\frac{v_1 + v_2 + |v_1 - v_2|}{2}$$

.

- O operador matemático de módulo ($|x|$) é representado em C pela função fabs(x)

5) Faça um programa que leia 2 valores e escreva o menor deles.

Dica: O menor entre dois valores pode ser obtido a partir da soma dos dois e do valor do maior.

6) Faça um programa que leia 3 valores escreva o maior deles.

Dica: O maior entre três valores pode ser encontrado pela aplicação repetida da expressão para encontrar o maior entre dois valores.

Operador de resto da divisão inteira

- Além dos operadores vistos (+,-,*,/), a linguagem C possui também o operador %, que resulta no RESTO da divisão inteira do primeiro operando pelo segundo. Ex: x = 5 % 2; x receberá 1, que é o resto da divisão inteira de 5 por 2.

- 7)Faça um programa que lê um valor inteiro em reais e calcula e escreve qual o menor número possível de notas de 100,50,20,10,5,2 e 1 real em que o valor pode ser decomposto. Dica:Isso pode ser calculado a partir de operações de divisão inteira.

	375 reais	100
		3 notas de 100
e sobra	75 reais	50
		1 nota de 50
e sobra	25 reais	20
		1 nota de 20
e sobra	5 reais	10
		0 notas de 10
e sobra	5 reais	5
		1 nota de 5
e sobra	0 reais	2
		0 notas de 2
e sobra	0 notas de 1	

```
leia(valor)
Notas100 <- valor \ 100
valor <- valor % 100
Notas50 <- valor \50
valor <- valor %50
...
```

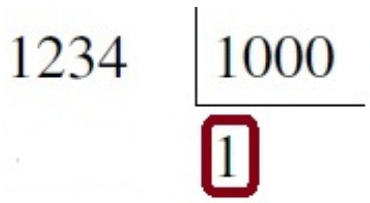
- 8) Faça um programa que lê uma quantia inteira em segundos e escreva o número de horas, minutos e segundos correspondente.
- 9) Faça um programa que lê 3 valores **a**, **b** e **c**, lados de um triângulo, e calcule e escreva a área do triângulo formado. A área de um triângulo de lados **a**, **b** e **c** pode ser calculada pela expressão

$$Area = \sqrt{S * (S - a) * (S - b) * (S - c)}$$

- onde S é o semi-perímetro, ou seja, a metade da soma dos lados ($\frac{a+b+c}{2}$).

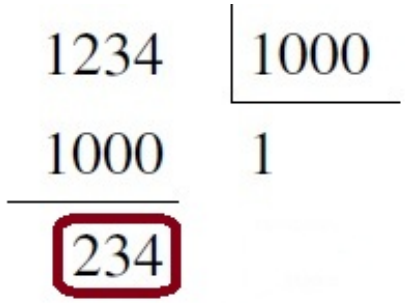
- 10) Faça um programa que le um valor entre 0 e 9999 e calcula a soma dos seus dígitos. O dígito menos significativo de um número inteiro pode ser obtido pelo resto da divisão do número por 10. Os dígitos restantes podem ser obtidos pela divisão inteira por 10.

Ex: Separação de dígitos



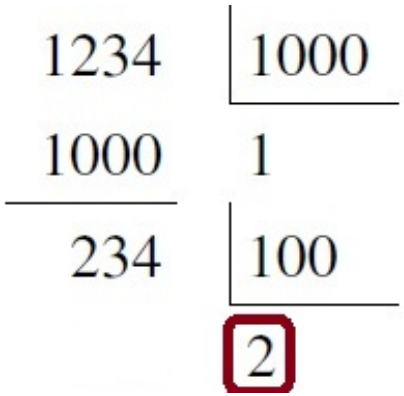

```
leia(valor)
milhar <- valor \ 1000
```

Ex: Separação de dígitos



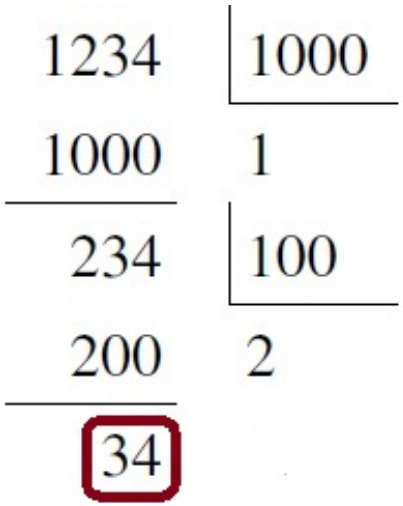

```
leia(valor)
milhar <- valor \ 1000
valor <- valor % 1000
```

Ex: Separação de dígitos



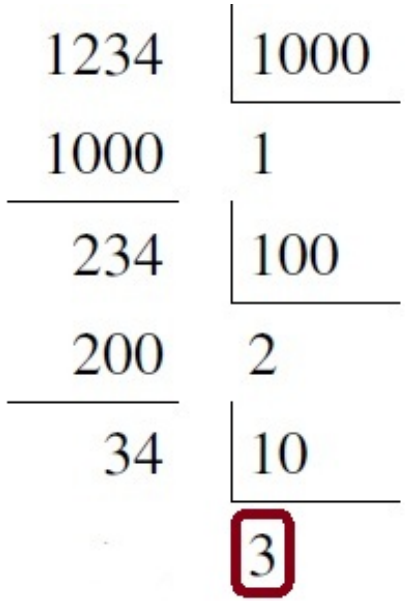

```
leia(valor)
milhar <- valor \ 1000
valor <- valor % 1000
centena <- valor \ 100
```

Ex: Separação de dígitos



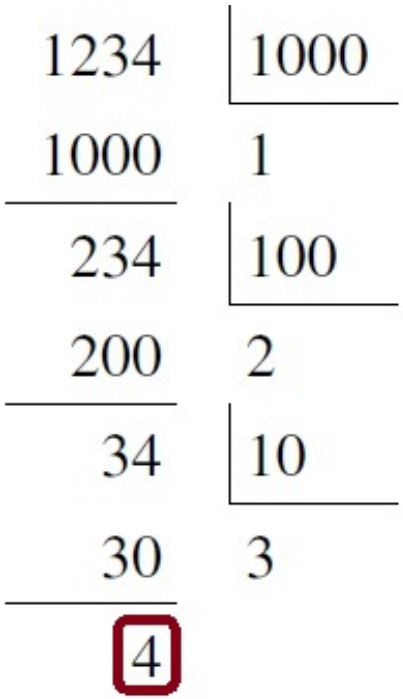

```
leia(valor)
milhar <- valor \ 1000
valor <- valor % 1000
centena <- valor \ 100
valor <- valor % 100
```

Ex: Separação de dígitos



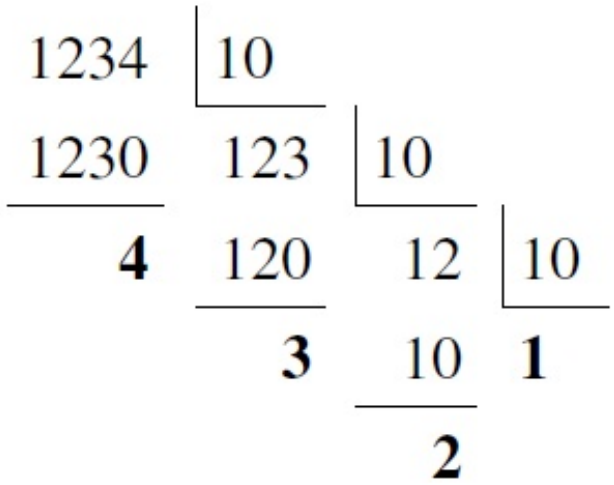

```
leia(valor)
milhar <- valor \ 1000
valor <- valor % 1000
centena <- valor \ 100
valor <- valor % 100
dezena <- valor \ 10
```

Ex: Separação de dígitos




```
leia(valor)
milhar <- valor \ 1000
valor <- valor % 1000
centena <- valor \ 100
valor <- valor % 100
dezena <- valor \ 10
unidade <- valor % 10
```

Ex: Separação de dígitos (b)




```
leia(valor)
unidade <- valor % 10
valor <- valor \ 10
dezena <- valor % 10
valor <- valor \ 10
centena <- valor % 10
milhar <- valor \ 10
```

- 11) Faça um programa que leia 4 valores, H_i, M_i, H_f, M_f, representando respectivamente a hora e minuto inicial e final de um evento, e calcule a duração do mesmo em horas e minutos. Considere que o evento inicia e termina no mesmo dia. Para simplificar o problema, converta cada par de valores em um único valor em minutos.

- 12) Faça um programa que leia dois horários (hora, minuto e segundo) e escreva quantos segundos transcorreram entre esses dois horários.
- 13) Faça um programa que a partir de um horário (hora, minuto, segundo) e uma quantidade de segundos transcorridos, calcule o segundo horário.
- 14) Faça um programa que leia a quantidade de alunos em uma sala de aula e a quantidade de alunos por grupo, e calcule e escreva quantos grupos serão formados e o resto de alunos que não foram suficientes para formar mais um grupo.

Biblioteca Locale.h

- Mensagens emitidas pelo printf não reconhecem caracteres acentuados.
- Para escrever mensagens com caracteres acentuados deve-se configurar o conjunto de caracteres a ser utilizado como os caracteres da língua portuguesa.
- Isso é feito no início do main pela chamada de função:

  setlocale (LC_ALL, "Portuguese");
- A função setlocale se encontra na biblioteca locale.h, é para usá-la é necessário o include

  #include <locale.h>

Comando If ou Comando Condicional

É utilizado quando o programa deve executar comandos diferentes em diferentes situações. Seu formato é:

```
if (condição)
{
comandos;
}

ou

if (condição)
{
comandos;
}
else
{
comandos;
}
```

onde **condição** é uma expressão de comparação que resulta verdadeiro ou falso.

- Os operadores de comparação são:
  - Igual : representado por $==$
  - diferente : representado por $!=$
  - maior, menor, maior ou igual, menor ou igual: $>$, $<$, $>=$, $<=$
- uma condição pode conter diversas expressões de comparação combinadas com o uso dos operadores && (e/and) e || (ou/or).
- Ex: if (a>0 && b>0)... if (a>0 || b>0)...
- uma expressão com && é verdadeira se ambas as subexpressões são verdadeiras.
- Uma expressão com || é verdadeira se pelo menos uma das expressões é verdadeira.

Exemplo:

```
if (a >= 0)
        printf("Positivo\n");
else
        printf("Negativo");
if (a > b)
        printf("O maior é %d\n",a);
else
        printf("O maior é %d\n",b);
```

- Se a lista de comandos do if ou do else tiver mais de um comando, eles devem ser agrupados utilizando chaves;
- Se a lista do if ou do else tiver apenas um comando, as chaves são desnecessárias (mas podem ser utilizadas mesmo com um único comando):

```
if (a > 0)
{
        printf("Sou positivo");
        b=0;
}
```

- O comando condicional deve ser utilizado quando há diversas situações que devem ser tratadas de formas diferentes pelo programa. Os passos para utilizá-lo são:
  1. Identificar com clareza quais as diferentes situações que devem ser tratadas pelo algoritmo.
  2. Definir qual o comportamento (comandos que devem ser executados) que o programa deve ter em cada uma das situações.
  3. Encontrar expressões que permitam ao programa identificar cada uma das situações.

Exercícios:

1) Faça um programa que leia um valor e escreva se o valor é igual a zero, se é maior que zero ou se é menor que zero.

2) Faça um programa que leia um valor inteiro e escreva se ele é par ou é ímpar.

3) Faça um programa que leia dois valores e, através de uma comparação, escreva o maior deles. O que deve ser feito se os dois valores são iguais?

4) Faça um programa que leia dois valores e escreva os dois em ordem crescente.
5) Faça um programa que leia 3 valores v_1, v_2 e v_3, e escreva-os em ordem crescente.
6) Faça um programa que leia 3 valores v_1, v_2 e v_3 e coloque-os em ordem crescente, de forma que v_1 contenha o menor, v_2 contenha o elemento do meio (nem o maior, nem o menor), e v_3 contenha o maior. Escreva os valores ordenados.

Tabela de precedência de operadores em C

Prec.	Símbolo	Função	Associatividade
1	++ - -	Incremento e decremento pósfixo	esquerda para direita
1	()	Parênteses (chamada de função)	
1	[ ]	Elemento de array	
1	.	Seleção de campo de structure	
1	->	Seleção de campo de structure a partir de ponteiro	
2	++ - -	Incremento e decremento prefixo	direita para a esquerda
2	+ -	Adição e subtração unária	
2	! ~	Não lógico e complemento	
2	(tipo)	Conversão de tipo de dado	
2	*	Desreferência	
2	&	Referência (endereço de elemento)	
3	* / %	Multiplicação, divisão, e módulo (resto)	esquerda para a direita
4	+ -	Adição e subtração	
5	<< >>	Deslocamento de bits à esquerda e à direita	
6	< <=	"menor que" e "menor ou igual que"	
6	> >=	"maior que" e "maior ou igual que"	
7	== !=	"Igual a" e "diferente de"	
8	&	E bit a bit	
9	^	Ou exclusivo para bits	
10	|	Ou para bits	
11	&&	E lógico	
12	||	Ou lógico	
13	c ? t : f	Condição ternária	direita para esquerda
14	=	Atribuição	
14	+= -=	Atribuição por adição ou subtração	
14	*= /= %=	Atribuição por multiplicação, divisão ou módulo (resto)	
14	<<= >>=	Atribuição por deslocamento de bits	
14	&= ^= |=	Atribuição por operações lógicas	
15	,	vírgula	esquerda para direita

7) Escreva um programa que leia os valores das quatro provas de um aluno e escreva a média aritmética considerando apenas as três melhores notas. Por exemplo, se o valores lidos foram 9, 9.5, 7, e 8, a média será $(9 + 9.5 + 8)/3$ (a prova de nota 7 é descartada). Não esqueça de considerar a possibilidade de ocorrerem notas iguais.

8) Faça um programa que leia 3 valores a, b e c, coeficientes de uma equação de segundo grau, e verifique se a equação tem raízes reais.

Se a equação tiver raízes reais, calcule e escreva as raizes da equação.
Se não tiver, escreva "A equação não possui raízes reais".
Dica:As raízes de uma equação podem ser calculadas pela fórmula de Baskhara.

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

- Uma equação não possui raízes se reais se $b^2 - 4ac < 0$

9) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele pode entrar em um dia e sair no outro, mas que o total de horas trabalhadas não excede 23 horas.

10) Faça um programa que leia 3 valores a, b e c, lados de um triangulo, e verifique o tipo de triangulo formado escrevendo:

0 - se o triângulo é equilatero (os três lados são iguais);

1 - se o triângulo é isósceles (dois lados iguais e um diferente);

2 - escaleno (3 lados diferentes).

11) Faça um programa que leia 3 valores a, b e c, lados de um triangulo, e verifique o tipo de triangulo formado escrevendo:

- 0 - se o triângulo é retângulo ($A^2 = B^2 + C^2$);
- 1 - se o triângulo é acutângulo ($A^2 > B^2 + C^2$) ;
- 2 - obtusângulo ($A^2 < B^2 + C^2$).

Considere que, para aplicar as relações mostradas, o programa deve garantir que o maior dos 3 lados estará em A.

12) Faça um programa que leia 3 valores l_1,l_2 e l_3 e verifique se formam um triângulo e, se formarem, o tipo de triângulo formado, escrevendo:

- 0 - se não formarem triângulo;
- 1 - se formarem um triângulo equilátero (os três lados são iguais);
- 2 - se formarem um triângulo isósceles (dois lados iguais e um diferente);
- 3 - se formarem um triângulo escaleno (3 lados diferentes)

- 13) Faça um programa que leia 3 notas de um aluno e escreva sua média harmônica.
- A média harmônica entre três valores N_1, N_2 e N_3 é calculada pela expressão

$$\frac{3}{\frac{1}{N_1} + \frac{1}{N_2} + \frac{1}{N_3}}$$

.

- O que acontece se alguma das notas for igual a 0? Que resultado o programa deve emitir?

14) Faça um programa que leia 3 notas de um aluno e escreva sua média harmônica. Se o aluno obteve média abaixo de 6.0, E SOMENTE NESSE CASO, leia uma quarta nota (da prova de recuperação) e substitua a menor das três notas pela nota da recuperação e recalcule a média harmônica. Escreva a média harmônica final e o conceito obtido:

- 0, se média harmônica (MH) $<$ 6.0;
- 1, se 6.0 $<=$ MH $<$7.0;
- 2, se 7.0 $<=$ MH $<$ 8.0;
- 3, se 8.0 $<=$ MH $<$ 9.0;
- 4, se MH$>=$9.0.

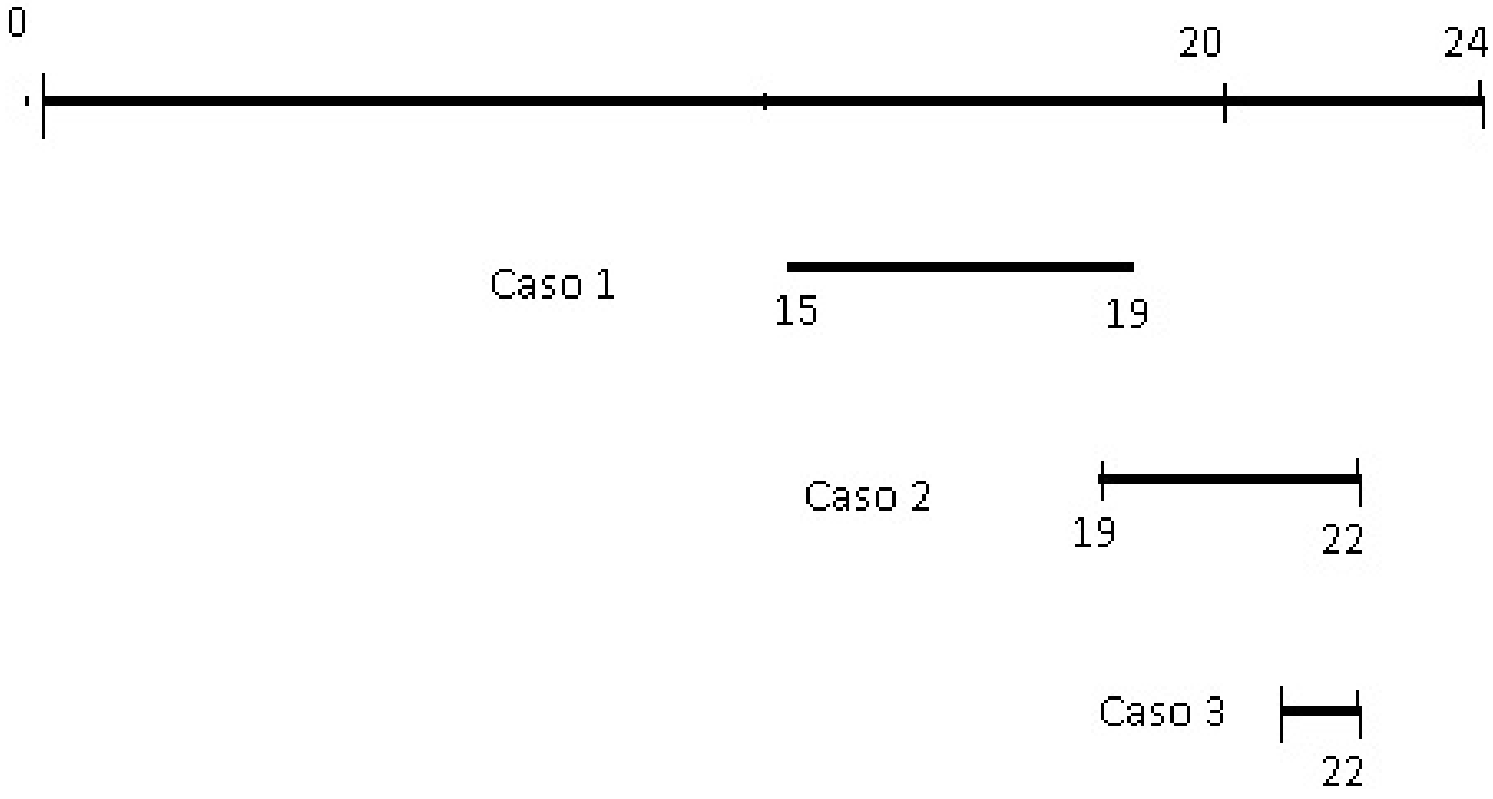
A média harmônica entre três valores N1, N2 e N3 é calculada pela expressão 3/(1/N1+1/N2+1/N3).

15) Faça um programa que leia 4(quatro) valores e escreva os 3 (tres) maiores em ordem decrescente. Considere que podem ocorrer valores iguais.

16) Faça um programa que leia 4 valores, Hi, Mi, Hf, Mf, representando respectivamente a hora e minuto inicial e final de um evento, e calcule a duração do mesmo em horas e minutos. Considere que o evento pode iniciar em um dia e terminar no dia seguinte. Algumas (mas não todas) situações que podem ocorrer são:

- Entrada: 10:30 Saída: 10:45
- Entrada: 10:30 Saída: 11:15
- Entrada: 22:15 Saída: 1:45
- Entrada: 22:15 Saída: 10:45

- 17) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele entra e sai no mesmo dia, e que as horas a partir das 20:00 valem 20% a mais (adicional noturno).
  - Quantos casos diferentes existem?
  - Que cálculo deve ser feito em cada caso?
  - Como o algoritmo pode identificar cada caso?


0
20
24
Caso 1
15
19
Caso 2
19
22
Caso 3
22

- 18) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele entra e sai no mesmo dia, e que as horas antes das 6:00 da manhã e a partir das 20:00 valem 20% a mais (adicional noturno).
  - Quantos casos diferentes existem?
  - Que cálculo deve ser feito em cada caso?
  - Como o algoritmo pode identificar cada caso?

- 19) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele pode entrar em um dia e sair no outro, mas que o total de horas trabalhadas não excede 23 horas.
  - Quantos casos diferentes existem?
  - Que cálculo deve ser feito em cada caso?
  - Como o algoritmo pode identificar cada caso?

- 20) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele pode entrar em um dia e sair no dia seguinte, e que se ele permanecer mais do que 8 horas, as duas horas a partir da nona hora valem 20% a mais, e as horas a partir da décima primeira hora valem 50% a mais (horas extras).

- 21) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele pode entrar em um dia e sair no outro, mas que o total de horas trabalhadas não excede 23 horas. Considere que as horas a partir das 20:00 valem 20% a mais (adicional noturno).

- 22) Faça um programa que leia para um trabalhador o valor que ganha por hora, a hora de entrada e a hora de saída (valores inteiros, sem minutos) e calcule quanto ele ganhou pelo turno. Considere que ele pode entrar em um dia e sair no outro, mas que o total de horas trabalhadas não excede 23 horas. Considere também que as horas antes das 6:00 da manhã e a partir das 20:00 valem 20% a mais (adicional noturno).

- 23) Escreva um prorama que leia duas datas, cada uma composta de Dia, Mês e Ano, e as escreva em ordem cronológica crescente. Ex:se as datas são 01/04/2000 e 17/05/1988, o algoritmo deve escrever 17/05/1988 01/04/2000.

- 24) A distância entre dois pontos definidos pelas coordenadas (X_1, Y_1) e (X_2, Y_2) é dada por

$$\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$$

.

- Faça um programa que leia 8 valores representando as coordenadas X e Y de 4 pontos,vértices de um polígono, e verifique se os pontos formam um quadrado, escrevendo:

  1 - se formam um quadrado;
  0 - se não formam.

- Considere que os pontos são lidos no sentido horário, seguindo o perímetro do quadrado.

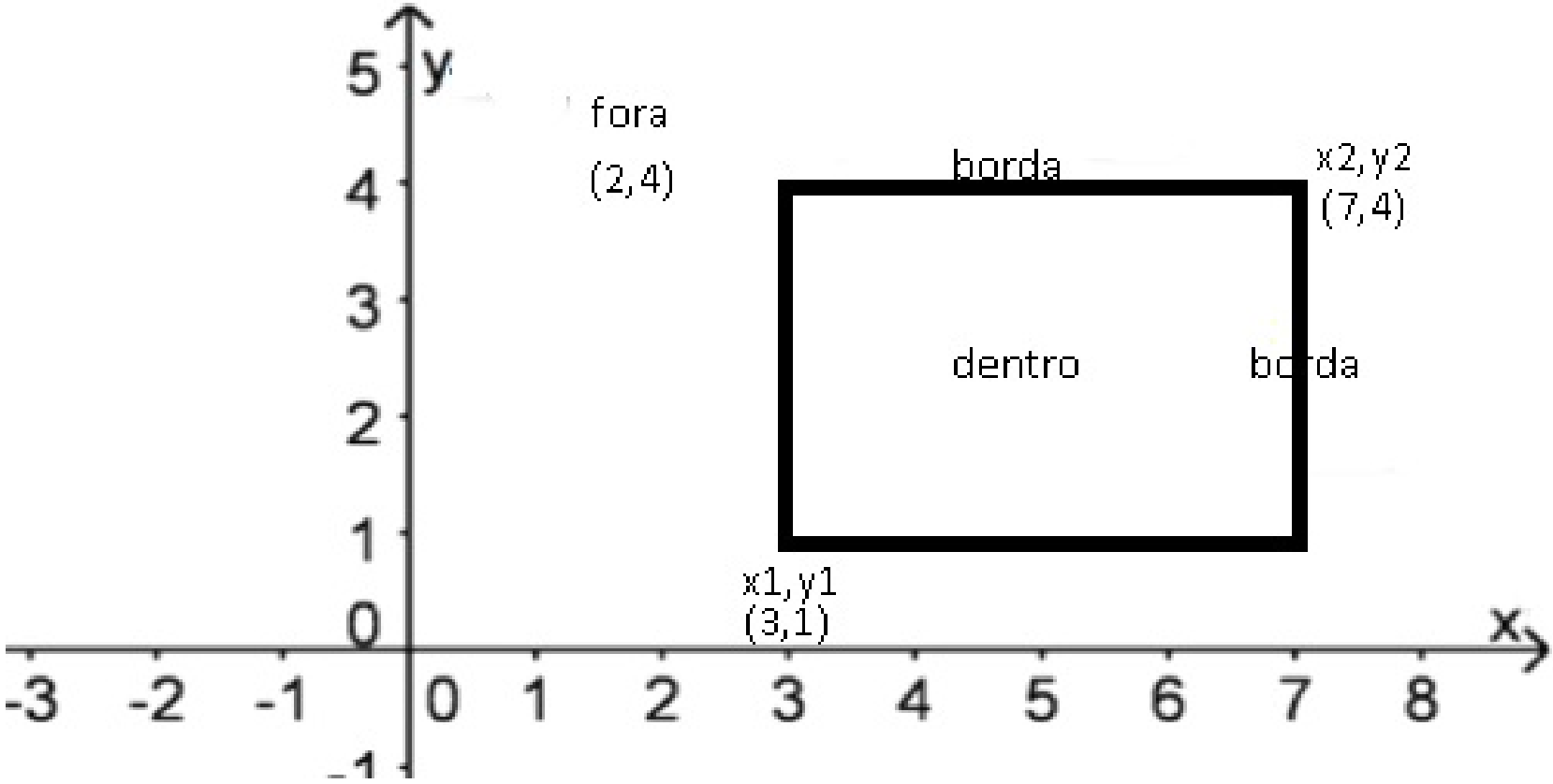
- 25) Faça um programa que leia oito valores correspondentes às coordenadas dos quatro vértices de um quadrilátero convexo no espaço cartesiano $(X_0, Y_0, X_1, Y_1, X_2, Y_2, X_3, Y_3)$. O algoritmo deve identificar o tipo de polígono formado escrevendo:
  1. se os 4 pontos formam um quadrado;
  2. se formam um retângulo;
  3. se formam um losango;
  4. se formam um paralelograma;
  5. se formam um papagaio (2 pares de lados adjacentes iguais. Ex: lados de tamanhos 3,3,4 e 4);
  6. se não formam nenhum dos anteriores.

- A distância (tamanho dos lados) entre dois pontos quaisquer (X_i, Y_i) e (X_j, Y_j) pode ser obtida através da fórmula distXiYiXjYj $= \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2)}$.
- Por exemplo, se os pontos lidos foram (3,2), (0,5), (3,8) e (6,5), a figura formada é um quadrado e o algoritmo escreve 1.
- Para que a figura seja um quadrado, os comprimentos das duas diagonais devem ser iguais, bem como os 4 lados.
- Se os pontos lidos foram (4,2), (1,4), (4,6) e (7,4), a figura formada é um losango.
- Se os pontos lidos foram (2,3), (0,5), (3,8) e (5,6), a figura formada é um retângulo.
- Se os pontos lidos foram (7,3), (0,3), (2,5) e (5,5), a figura formada não é nenhuma das anteriores e o programa escreve 6
- Os pontos são fornecidos em sentido horário ao longo do perímetro do quadrilátero.

- 26) Faça um programa que leia as dimensões (altura, largura e profundidade) de duas caixas e verifique se a primeira caixa pode ser colocada dentro da segunda. Considere que as caixas podem ser rotacionadas em qualquer direção. Se a primeira caixa couber dentro da segunda escreva 1, caso contrário escreva 0.

- 27) Faça um programa que leia 4 valores X1,Y1,X2,Y2, correspondendo às coordenadas do canto inferior esquerdo e canto superior direito de uma região retangular no plano. Leia a seguir dois valores X,Y correspondendo a um ponto no plano e escreva:

  0 - Se o ponto está fora da região retangular;
  1 - Se o ponto está dentro da região retangular;
  2 - Se o ponto está exatamente na borda da região retangular.


y
5
4
3
2
1
0
-1
fora
(2,4)
borda
x2,y2
(7,4)
dentro
borda
x1,y1
(3,1)
x
-3
-2
-1
0
1
2
3
4
5
6
7
8

- 28) Escreva um programa que leia duas datas, cada uma composta de Dia, Mês e Ano, e as escreva em ordem cronológica crescente.
- Ex:se as datas são 01/04/2000 e 17/05/1988, o algoritmo deve escrever 17/05/1988 01/04/2000.
- Sugestões:
  1. Converta cada data (dia,mês,ano) em um único valor, da forma como fizemos com hora-minuto-segundo, para poder efetuar uma única comparação; ou
  2. Efetue primeiro a comparação entre os anos, se esses forem iguais compare os meses e, se anos e meses forem iguais compare os dias.

- 29) Escreva um programa que leia três datas, cada uma composta de Dia, Mês e Ano, e as escreva em ordem cronológica crescente.
- Ex:se as datas são 01/04/2000, 17/05/1988 e 23/10/1969, o algoritmo deve escrever 23/10/1969 17/05/1988 01/04/2000.
- Sugestão: As mesmas do exercício anterior

- 30) Escreva um programa que leia uma data, composta por dia, mês e ano, e verifique se a data corresponde ao último dia do mês, escrevendo:
  - 1, se for o último dia do mês
  - 0 se não for o último dia do mês.
- Considere, para simplificar o problema, que ano bissexto é aquele divisível por 4, e que fevereiro tem 28 dias (29 em ano bissexto).
- Setembro, abril, junho e novembro têm 30 dias e todos os outros meses tem 31 dias.

- 31) Escreva um algoritmo que leia uma data, composta por dia, mês e ano, e escreva a data correspondente ao dia seguinte.
- Considere, para simplificar o problema, que ano bissexto é aquele divisível por 4, e que fevereiro tem 28 dias (29 em ano bissexto), setembro, abril, junho e novembro têm 30 dias e todos os outros meses tem 31 dias. Sugestão: lâmina seguinte

leia(dia,mes,ano)
Se {dia é o último dia do ano}
    estao escreva (1, 1, ano+1)
senao se {dia é o último dia do mes}
    estao escreva (1, mes+1, ano)
senao escreva(dia+1,mes,ano)

32) Escreva um programa que leia uma data, composta por dia, mês e ano, e escreva quantos dias passaram-se desde o início do ano. Considere, para simplificar o problema, que ano bissexto é aquele divisível por 4, e que fevereiro tem 28 dias (29 em ano bissexto), setembro, abril, junho e novembro têm 30 dias e todos os outros meses tem 31 dias.

33) Para enviar uma carta são necessários um selo e um envelope. O selo custa 12 centavos e o envelope custa 5 centavos. Faça um programa que leia uma quantia inicial de selos, envelopes e centavos, e escreva o número de cartas que podem ser enviadas com esses selos, envelopes e centavos (utilizando-os para comprar mais selos e envelopes). Considere que não é possível converter selos ou envelopes em dinheiro.

34) Para enviar uma carta são necessários um selo e um envelope. O selo custa 12 centavos e o envelope custa 5 centavos. Faça um programa que leia uma quantia inicial de selos, envelopes e dinheiro (em reais), e escreva o número de cartas que podem ser enviadas com esses selos, envelopes e centavos (utilizando-os para comprar mais selos e envelopes). Escreva também, nessa ordem, a quantidade de selos, envelopes e dinheiro (em centavos), que restará após enviadas as cartas. Considere que não é possível converter selos ou envelopes em dinheiro.

35) Uma fábrica produz um recipiente de plástico com sua tampa (também de plástico). Ambos os componentes utilizam o mesmo equipamento para fabricação (ou seja, não podem ser fabricados ao mesmo tempo). A fabricação do recipiente consome duas horas; a fabricação da tampa consome meia hora. Um cliente deseja o máximo de recipientes (com tampa) para 10 dias. A fábrica trabalha 24 horas/dia e já dispõe de uma quantidade r de recipientes e t de tampas em seu estoque (não necessariamente iguais). Faça um programa que leia os valores de r e t e informe o máximo de conjuntos recipiente-tampa que ela pode fornecer em 10 dias.

36) Escreva um programa que leia dois valores D e DS, correspondentes a um dia do mes, e ao dia da semana que corresponde a ele (1-domingo 2-segunda 3- terça 4-quarta 5-quinta 6-sexta 7-sábado). Calcule e escreva em que dia da semana caiu o dia primeiro do mês do dia digitado.
Exemplo: dia 10 no mês é 3 (terça) na semana. Resposta 1 (domingo)
Exemplo: dia 30 no mês é 4 (quarta) na semana. Resposta 3 (terça feira)

37) Faça um programa que leia as dimensões (altura, largura e profundidade) de duas caixas e verifique se a primeira caixa pode ser colocada dentro da segunda. Considere que as caixas podem ser rotacionadas em qualquer direção. Se a primeira caixa couber dentro da segunda escreva 1, caso contrário escreva 0.

38) Faça um programa que leia dois números de três dígitos cada um, e verifique se possuem os mesmos dígitos. Considere que pode haver dígitos repetidos em um número, e que a cada dígito de um número deve corresponder exatamente um dígito do outro número. Assim, os números 221 e 112 não tem os mesmos dígitos, porque apesar de ambos terem somente os dígitos 1 e 2, aos dois dígitos 2 do primeiro número corresponde o mesmo dígito no segundo número. O algoritmo deve escrever 1, caso os números tenham os mesmos dígitos, e 0 em caso contrário.

39) Faça um programa que leia um número de três dígitos e escreva o maior número que possui os mesmos dígitos do número lido. Se um dígito aparecer repetido no número lido, deve ser repetido o mesmo número de vezes no número gerado.

Comandos de Repetição

- São utilizados para que um comando ou conjunto de comandos sejam executados mais de uma vez.
- A linguagem C possui 3 comandos de repetição:
  - O comando **for**
  - O comando **while**
  - O comando **do**...**while**
- Começaremos vendo o comando **for**, já que, de modo geral, é o mais usado em C.

Comando **for**

- O comando for é um comando de repetição do tipo "enquanto", que contem a inicialização e o incremento da variável de controle.
- Sua sintaxe é:

```
for (inicialização; condição; incremento)
    comando;
```

- onde **comando** é um comando simples ou um grupo de comandos, agrupados com chaves.
- **Inicialização** é um conjunto de atribuições iniciais (pode ser mais de uma, separadas por vírgulas).
- **Condição** é a expressão lógica que, enquanto for verdadeira, faz com que o comando continuará sendo executado.
- **Incremento** é um conjunto de comandos que será executado ao final de cada iteração.

Equivalência entre o "enquanto" e o "for"

```
i ← 0
enquanto i<10 faca
        comando
        i←i+1
fimenquanto
```

```
for (i=0;i<10;i=i+1)
        comando;
```

Observações

- Os campos de inicializações, condição e incrementos são separados entre si por ponto-e-vírgula.
- O campo **inicializações** do cabeçalho do for pode conter mais de uma inicialização.
  - Nesse caso, as inicializações são separadas por vírgulas.
  - O mesmo ocorre nos incrementos.
  - Ex: for (i=0,j=0; j<10; j=j+1,i=i+1)...
- O cabeçalho do for não precisa obrigatoriamente conter inicializações ou incrementos.
  - Nesse caso as listas de inicializações e incrementos ficam vazias, mas os ponto-e-vírgula entre os campos de inicializações, condição e incrementos são obrigatórios.
  - Ex: for (;i<10;) ...

Ex:

```
for (i=0 ; i<10 ; i=i+1)
/* para i de 0, ENQUANTO i for menor que 10, faça */
      printf ( "%d \n " , i ) ;
```

Ex. com dois comandos de inicialização e dois de incremento:

```
for (i=10 , j=1 ; i<10 ; i=i+1 , j=j-1 )
      printf ( "i = %d j = %d \n " , i , j ) ;
```

Ex. com mais de um comando na lista de comandos:

```
for (soma=0 , i=0 ; i<10 ; i++)
      {
      soma=soma+i;
      printf("soma = %d \n",soma);
      }
```

- Cada repetição dos comandos dentro do comando for também é chamada **iteração** (daí o nome de "**programas iterativos**").
- Erros comuns:

1) Colocar um ponto-e-vírgula na linha do for:

```
for (i=0;i<10;i=i+1);
        printf("%d\n",i);
```

Assim como no **if**, esse ponto-e-vírgula faz com que o comando **for** termine, de modo que o comando seguinte está fora do **for** e será executado só uma vez.

2) Não colocar chaves no grupo de comandos:

```
for (i=0,soma=0;i<10;i=i+1)
    soma=soma+i;
    printf("soma=%d\n",soma);
```

Nesse exemplo, apenas o primeiro comando (soma=soma+i) está dentro do **for** e será repetido.

3) Tratar a condição do **for** como uma condição de repita (até que...) ao invés de uma condição de enquanto. Ex.

```
for (i=0;i==10;i=i+1)
// pensando que a condição é ATÉ QUE i=10
    printf("%d\n",i);
```

Pós-incremento e Pré-incremento

- i = i +1 pode ser escrito como i++ ou ++i
- i = i -1 pode ser escrito como i- - ou - -i
- A diferença entre o ++i (pré-incremento) e o i++ (pós incremento) é que, quando ocorre dentro de uma expressão, o valor utilizado na expressão é o valor antes do incremento ou depois do incremento.
- x=i++; // o incremento de i é feito após a atribuição
- x=++i; // o incremento de i é feito antes da atribuição

Formas reduzidas de atribuição:

- Quando um dos operandos de uma operação é a variável que receberá o resultado (p.ex: a = a + 3;) pode-se omitir o operando da expressão, resultando o comando a+=3;
- Da mesma forma pode-se utilizar essa versão reduzida para os operadores de subtração, divisão e multiplicação:
  - a*=3; equivale a a=a*3;
  - a-=3; a=a-3;
  - a/=3; a=a/3;
- de forma que uma forma de fazer um laço para escrever os números de 1 a 10 fica:

```
for ( i = 1 ; i <= 10 ; i+=1 )
    printf("%d\n",i);
```

Comandos break e continue:

- O comando **break** é utilizado quando se deseja interromper a execução das repetições, sem completá-las
- A execução do comando **break** faz com que a execução vá para o comando seguinte ao comando **for**, sem completar a iteração corrente.
- ex:

```
for (i=0;i<10;i++)
    {
    scanf("%d",&a);
    if (a==0) break;
    /* faz com que saia do for se for digitado 0 */
    }
```

Comando Continue

- O comando **continue** é utilizado quando se deseja interromper a execução da iteração atual, mas deseja-se continuar executando as próximas iterações do for.

Uso de valores numéricos como valores lógicos (verdadeiro, falso):

- A linguagem C permite utilizar valores numéricos como valores lógicos (verdadeiro ou falso).
- Nesse sentido, o valor 0 representa o valor lógico falso, e um valor diferente de 0 representa o valor lógico verdadeiro.
- Assim o for a seguir faz com que o programa fique indefinidamente dentro do for, até ser executado um comando break:

```
for (;1;)
{
/* comandos */
}
```

A função kbhit( ):

A função kbhit( ), declarada na biblioteca conio.h (#include <conio.h>) verifica se algum valor foi teclado, retornando verdadeiro se algum valor foi teclado, e retornando falso se nenhum valor foi teclado.

```
#include <stdio.h>
#include <conio.h>
int main()
{
     int i=0;
     for (;1;)
     {
          if (kbhit()) break;
          printf("%d\n",i++);
     }
}
```

Exercícios:

- 1) Faça um programa que escreva os números de 1 a 100.
- 2) Faça um programa que escreva todos os números pares entre 1 e 50.
- 3) Faça um programa que escreva todos os números entre 1 e 200 que são múltiplos de 11.
- 4) Faça um programa que leia 10 números e escreva os que forem pares. Dica: Pode-se verificar se um número é par verificando-se se o resto de sua divisão inteira por 2 é igual a 0.
- 5) Faça um programa que leia números até que seja digitado um número negativo, e escreva todos que forem pares.

6) Faça um programa que leia 10 valores e conte quantos estão no intervalo [10,20], escrevendo ao final essa informação.

- 7) Faça um programa que leia 100 valores e conte quantos estão em cada um dos intervalos [0,25], (25,50], (50,75], (75..100], escrevendo ao final essa informação.
  - Lembrando que o colchete representa um intervalo fechado (isto é, que inclui o limite do intervalo) e os parênteses representam um intervalo aberto (que não incluem o limite do intervalo)

8) Faça um programa que leia um valor N. Escreva a seguir os números de 1 a N.
9) Faça um programa que leia um valor N. Leia, a seguir, N valores e escreva todos que forem positivos.

10) Faça um programa que leia um valor N, e escreva todos os seus divisores.
11) Faça um programa que leia um valor N, e conte quantos divisores possui, escrevendo ao final essa informação.
12) Faça um programa que leia um valor N, e calcule e escreva a soma dos seus divisores.

13) Faça um programa que leia um número N e escreva seu primeiro divisor maior do que 1.
14) Um número inteiro maior do que 1 é primo se ele possui como divisores somente o 1 e ele mesmo. Faça um programa que leia um número e verifique se é primo escrevendo uma mensagem apropriada. Dica: Pode-se também verificar se um número é primo encontrando seu primeiro divisor maior que 1. Se o primeiro divisor for o próprio número, ele é primo.

15) Dois números n_1 e n_2 são ditos amigos entre si se a soma dos divisores de n_1 (excluindo o próprio n_1) é igual a n_2, e a soma dos divisores de n_2 (excluindo o próprio n_2) é igual a n_1. Ex: 220 e 284. Faça um programa que leia 2 valores e verifique se são amigos entre si escrevendo uma mensagem apropriada.

16) Faça um programa que leia 10 números positivos e escreva ao final o maior deles.

17) Faça um programa que leia 10 números, positivos ou negativos, e escreva ao final o maior deles.

18) Faça um programa que leia 10 números e escreva ao final o maior e o menor deles.
19) Faça um programa que leia, para 10 pessoas, seu peso e altura e escreva o peso e a altura da pessoa mais alta.
20) O índice de massa corporal (IMC) de uma pessoa é dada pela relação peso/($altura^2$). Faça um programa que leia, para 10 pessoas, seu peso e altura e escreva o IMC, peso e altura da pessoa de maior IMC.

Comando while

- Tem o formato

```
while (condição)
    comando
```

ou, se houver mais de um comando,

```
while (condição)
    {
    comandos
    }
```

- onde **condição** é uma condição de enquanto.
- Funciona como um **for**, sem a inicialização e o incremento.

comando do...while

- tem o formato

```
do {
        comandos
} while (condição);
```

- e **condição** é uma condição de enquanto.
- A diferença em relação ao **for** e ao **while** é que a condição é avaliada após a execução dos comandos, de modo que os comandos são executados ao menos uma vez.
- Cabe salientar que, apesar de a condição ser avaliada ao final, como no **Repita**, ainda é um **Enquanto** e a condição é uma condição para continuar executando, e não uma condição de saída
- Ex:

```
i=1;
do {
      printf("%d\n",i);
      i=i+1;}
while (i<=10);
```

1) Faça um programa que leia números até ser digitado um número negativo e escreva ao final a sua soma (o número negativo não entra na soma). Faça uma versão com o while e uma com o do...while, com e sem o uso do comando break.
2) Faça um programa que leia um valor N inteiro e maior do que 1, e calcule e escreva o seu menor divisor maior do que 1.

3) Faça um programa que leia 10 números e escreva os que forem primos.
4) Faça um programa que escreva os 50 primeiros números primos.
5) Faça um programa que leia 2 números N_1 e N_2 e escreva a soma dos números primos entre N_1 e N_2 (incluindo N_1 e N_2 se algum deles for primo).

7) Faça um programa que leia 2 números N_1 e N_2 e escreva o produto dos números primos entre N_1 e N_2 (incluindo N_1 e N_2 se algum deles for primo).
8) Faça um programa que leia um número N e escreva os N primeiros números primos maiores que 100.
9) Faça um programa que leia um número inteiro N e escreva o maior número primo menor do que N.

10) Um número perfeito é o número que é igual à soma de seus divisores, exceto o próprio número (ex: $6 = 1 + 2 + 3$. Faça um programa que leia 10 números e verifique para cada um se é perfeito ou não, escrevendo uma mensagem apropriada.
11) Faça um programa que leia, para um funcionários: o valor que ganha por hora e 30 pares de valores (hora de entrada e hora de saída, inteiros, sem minutos) e calcule e escreva o quanto ganhou no mês. O funcionário não pode trabalhar mais de 23 horas seguidas e pode iniciar em um dia e terminar no dia seguinte.

12) O MDC (máximo divisor comum) entre dois números n_1 e n_2 é o maior número que é divisor de n_1 e de n_2. Faça um programa que leia dois números e escreva seu MDC.
13) Faça um programa que leia 10 valores inteiros menores que 20 e, para cada um, calcule e escreva seu fatorial. O programa deve ignorar todos os valores maiores ou iguais a 20.
14) O MMC (mínimo múltiplo comum) entre dois números n_1 e n_2 é o menor número que é múltiplo de n_1 e de n_2. Faça um programa que leia dois números e escreva seu MMC.

15) Faça um programa que leia um número N e efetue sua fatoração, escrevendo os fatores que o compõem. Ex: 28 = 2 x 2 x 7 60 = 2 x 2 x 3 x 5
16) Faça um programa que leia 10 números e para cada um escreva a soma dos seus digitos formantes.

- Algoritmos de separação de dígitos de modo geral são baseados no cálculo do resto da divisão inteira por 10.
- Essa operação se repete até que não sobrem mais dígitos. Algo como:

```
Leia (N)
Soma ← 0
Enquanto N>0 faca
    Soma ← Soma + N%10
    N ← N \ 10
fimenquanto
Escreva(Soma)
```

17) Faça um programa que leia um número inteiro e escreva quantas vezes ocorre o dígito 2.

18) Faça um programa que leia um número N até 100.000.000 e verifique se possui dígitos repetidos escrevendo:

- 1 - se possuir dígitos repetidos;
- 0 - se não possuir.

19) Faça um programa que leia um número N até 100.000.000 e verifique se é palíndromo, ou seja, se se igual quando lido da esquerda para a direita e da direita para a esquerda. Ex: 13731. Escreva 1 se for palíndromo e 0 se não for.

20) Faça um programa que escreva os primeiros 100 dígitos cuja soma dos dígitos formantes é 10.
21) Uma pessoa aplicou um determinado valor em um banco. Sabe-se que o banco pagará 5% ao mês de juros. Faça um programa que leia o valor aplicado e calcule e escreva a quantidade mínima de meses necessários para que a pessoa obtenha R$1000,00 ou mais de rendimento. Por exemplo, se a pessoa aplicou 10.000 reais, ao final do primeiro mês terá 10.500, e ao final do segundo mês terá 11.025 e foram necessários 2 meses para alcançar um rendimento de mais de 1000 reais.

22) Faça um programa que leia 20 números inteiros e escreva quantos números são iguais ao menor número lido. Dica: Use um contador, incrementando-o ao encontrar um elemento igual ao menor corrente, e reinicializando-o ao encontrar um elemento menor do que o menor corrente.
23) Data juliana é o número de dias tanscorridos no ano (ex:236/1995). A faixa é de 1 a 365 (366 se o ano for bisexto). Escreva um programa algoritmo que leia uma data juliana (dia e ano) e a converta para o formato DD/MM/AAAA escrevendo a data. Considere que ano bisexto é aquele divisível por 4.

- 24) Faça um programa que leia duas datas, cada uma formada por dia, mês e ano, e escreva o número de dias entre as duas, incluindo a data inicial e a data final. Considere que ano bisexto é aquele divisível por 4.
- 25) Faça um algoritmo que leia duas datas, cada uma formada por dia, mês e ano, e escreva todas as datas entre as duas, incluindo a data inicial e a data final. Considere que ano bisexto é aquele divisível por 4. Considere também que a data inicial é menor ou igual à data final.

- 26) Faça um algoritmo que calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:
  1+3+5+7+9....
- 27) Faça um programa que calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:

$$\frac{1}{1} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{9} + ...$$

- 28) Faça um programa que leia um valor X e calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:

$$\frac{X}{1} + \frac{X}{3} + \frac{X}{5} + \frac{X}{7} + \frac{X}{9} + ...$$

- 29) Faça um programa que leia um valor X e calcule e escreva a soma dos 100 primeiros termos da sequência a seguir:

$$\frac{X}{1} - \frac{X}{3} + \frac{X}{5} - \frac{X}{7} + \frac{X}{9} - \ldots$$

- 30) Faça um programa que leia um valor N e escreva a soma dos N primeiros termos da série a seguir:

$$\frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{7} + \frac{1}{11} + \frac{1}{13} + \frac{1}{17} + \ldots$$

ou seja, a série onde os denominadores são os números primos

- 31) Um número piramidal é um número que é igual à soma de 3 números primos consecutivos (ex: $15 = 3 + 5 + 7$). Faça um programa que leia um valor N e escreva os 10 primeiros números piramidais maiores ou iguais a N.
- 32) Faça um programa que leia 30 números e escreva ao final:
  - a média dos números pares digitados (soma dos pares dividido pela quantidade de pares);
  - quantos números são primos e
  - o produto (multiplicação) de todos os números lidos que são múltiplos de 5.

33) Faça um programa que leia 20 números e verifique qual a maior sequência crescente dentre os números digitados, escrevendo ao final o primeiro e último valor da sequência.

- 34) Faça um programa que leia uma sequência de 20 números e escreva:

  0 - Se todos os números são iguais;
  1 - Se a sequencia é não-decrescente (cada número é maior OU IGUAL ao anterior);
  2 - se a sequencia é estritamente crescente (cada número é MAIOR que o anterior);
  3 - Se a sequencia é não-crescente (cada número é menor OU IGUAL ao anterior);
  4 - Se a sequencia é estritamente decrescente (cada número é MENOR que o anterior);
  5 - A sequencia é desordenada (há partes crescentes e partes decrescentes)

35) Faça um programa que leia a média e a quantidade de faltas para cada aluno de um grupo de 20 alunos, e escreva ao final o percentual de alunos aprovados. Os alunos serão considerados reprovados se não atingirem média 6.0. Também serão reprovados os alunos que tiverem mais de 20 faltas. Os alunos com uma quantidade de faltas entre 10 e 20 serão aprovados se obtiverem uma média mínima de 7.5.

- 36) Faça um programa que leia um número inteiro qualquer e escreva os dígito desse número em ordem crescente do valor de cada dígito.
- 37) Faça um programa que leia um número inteiro qualquer e verifique se possui algum dígito repetido escrevendo ao final:
  - 0 - se ele não contem nenhum dígito repetido;
  - 1 - se ele contem algum dígito repetido.

38) Faça um programa que leia o dia e mês de uma data, e o dia da semana (1 - domingo, 2 - segunda... 7 - sábado) e escreva o dia da semana correspondente ao dia primeiro do mês digitado.
39) Faça um programa que leia um número de 8 dígitos e calcule a soma ponderada dos seus dígitos, com peso 1 para o dígito menos significativo, peso 2 para o segundo dígito, peso 3 para o terceiro e assim por diante. Escreva a soma calculada.

40) Faça um programa que leia um número inteiro e escreva quantas vezes ocorre o dígito 2.
41) Faça um programa que leia um número inteiro (máximo 5 casas - não maior que 99999) e mostre quantos dígitos contem de cada número de 0 a 9
42) Faça um programa que leia um número inteiro (máximo 5 casas - não maior que 99999). Mostre o número que mais se repete. Em caso de empate, mostre o menor deles.

- 43) Um número inteiro é dito ascendente se cada um dos seus algaristmos é maior do que o algarismo imediatamente à sua esquerda. Por exemplo, o número 3589. Faça um programa que leia um número inteiro e verifique se ele é ou não ascendente escrevendo:
  - 1 - se ele é ascendente;
  - 0 - se ele não é ascendente.

44) Faça um programa que leia o dia e mês de uma data, e o dia da semana (1 - domingo, 2 - segunda... 7 - sábado) e escreva o dia da semana correspondente ao dia primeiro do mês digitado.
45) Faça um programa que leia um número de 8 dígitos e calcule a soma ponderada dos seus dígitos, com peso 1 para o dígito menos significativo, peso 2 para o segundo dígito, peso 3 para o terceiro e assim por diante. Escreva a soma calculada.

46) Faça um programa que leia 20 números e escreva a maior sequência de valores iguais entre os números digitados.
47) Faça um programa que leia um número inteiro positivo N não maior que 1000000000000000000 (1E18), calcule e escreva o maior inteiro menor ou igual a sua raiz quadrada, ou seja, calcule e escreva a parte inteira da raiz quadrada no número informado.

48) Georg Cantor demonstrou que os números racionais são enumeráveis pela sequência:

$$\frac{1}{1}\ \frac{1}{2}\ \frac{2}{1}\ \frac{1}{3}\ \frac{2}{2}\ \frac{3}{1}\ \frac{1}{4}\ \frac{2}{3}\ \frac{3}{2}\ \frac{4}{1}\ \frac{1}{5}\ \frac{2}{4}\ldots$$

Fazer um programa que leia N (no intervalo de 1 até 1000000) e escreva o enésimo número dessa sequência (escreva o numerador e o denominador).

49) Georg Cantor demonstrou que os números racionais são enumeráveis pela sequência:

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

Fazer um programa que leia N (no intervalo de 1 até $1000000000000000000 = 1E18$) e escreva o enésimo número dessa sequência (escreva o numerador e o denominador).

50) O CPF é formado por onze dígitos (999999999-99), dos quais os dois últimos são verificadores (controle), ou seja, a partir dos nove primeiros dígitos pode-se determinar os últimos dois. Considerando o CPF no formato abcdefghi-jk, onde cada letra representa um dígito, pode-se:

- calcular o primeiro dígito verificador (j), da seguinte forma:
    - somar: $10a + 9b + 8c + 7d + 6e + 5f + 4g + 3h + 2i$
    - encontrar o resto da divisão dessa soma por 11.
    - se o resto for igual a zero ou um, o dígito é zero, senão o dígito é onze menos esse resto.
- calcular o segundo dígito verificador (k):
    - somar: $11a + 10b + 9c + 8d + 7e + 6f + 5g + 4h + 3i + 2j$
    - encontrar o resto da divisão dessa soma por 11.
    - se o resto for igual a zero ou um, o dígito é zero, senão o dígito é onze menos esse resto.

Fazer um programa que leia o CPF (somente primeiros nove dígitos) e escreva separadamente os verificadores (dois últimos).

- 51) Foram entrevistados 500 alunos de uma universidade. Para cada aluno entrevistado foi registrado o código do curso que ele frequenta (1: engenharia; 2: computação; 3: administração) e sua idade. Faça um programa que processe tais dados fornecendo:
  - (a) número de alunos por curso;
  - (b) número de alunos com idade entre [20 25] anos por curso; e
  - (c) código do curso com a menor média de idade.

  Dica:São necessários vários contadores.

52) O quociente da operação de divisão pode ser obtido subtraindo-se o divisor do dividendo. Da diferença, subtraímos novamente o divisor e assim sucessivamente até que a diferença seja menor do que o divisor. A quantidade se subtrações é o quociente. Assim, por exemplo;
-21 /4 =
21 - 4 = 17 (1)
17 - 4 = 13 (2)
13 - 4 = 9 (3)
9 - 4 = 5 (4)
5 - 4 = 1 (5)
Para o exemplo acima, o número de subtrações é 5. Logo, o quociente é -5 (divisor e dividendo têm sinais diferentes). Faça um programa que leia o dividendo (0, positivo ou negativo) e o divisor (positivo ou negativo) e escreva o quociente usando o algoritmo acima. NÃO deve ser usado o operador de divisão. Dica: Registre, de alguma forma, os sinais dos operandos e transforme-os para positivo.

53) A operação de multiplicação, quando o multiplicador é um número inteiro, nada mais é do que uma sucessão de somas. Assim, por exemplo,
4X5= 5+5+5+5= 20.
Escreva um programa que leia o multiplicando e o multiplicador e, através de uma sucessão de somas, calcule e escreva o produto. O multiplicador é, necessariamente, um número inteiro. Porém, multiplicador e multiplicando podem ser 0 ou negativos.
17 - 4 = 13 (2)
13 - 4 = 9 (3)
9 - 4 = 5 (4)
5 - 4 = 1 (5)
Para o exemplo acima, o número de subtrações é 5. Logo, o quociente é -5 (divisor e dividendo têm sinais diferentes). Faça um algoritmo que leia o dividendo (0, positivo ou negativo) e o divisor (positivo ou negativo) e escreva o quociente usando o algoritmo acima. NÃO deve ser usado o operador de divisão. Dica: Registre, de alguma forma, os sinais dos operandos e transforme-os para positivo.

54) Um número N é dito um Meridiano Perfeito se existe um número M para o qual a soma dos números de 1 a N-1 é igual à soma dos números de N+1 a M. Assim, 6 é um Meridiano Perfeito porque 1+2+3+4+5=7+8. Faça um programa que leia um número e verifique se é um meridiano perfeito, escrevendo 1, se o número é um meridiano perfeito, e 0, se não for.

55) Um número N é dito um Meridiano Perfeito se existe um número M para o qual a soma dos números de 1 a N-1 é igual à soma dos números de N+1 a M. Assim, 6 é um Meridiano Perfeito porque 1+2+3+4+5=7+8. Faça um programa que leia um valor N, e escreva os N primeiros Meridianos Perfeitos maiores que 1.

56) Um número é dito regular, ou número de hamming (ou "5-smooth number" ou "ugly number") se tem como fatores primos apenas 2, 3 e 5, ou seja, é divisível somente por múltiplos de 2, 3 ou 5. Assim, os primeiros números regulares são 1 (é regular por definição, $2^0 * 3^0 * 5^0$), 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 27,30,32.... Faça um programa que leia um número N, menor ou igual a 100, e escreva os N primeiros números regulares.

57) Faça um programa que para 10 cartas de baralho leia o seu valor (entre 1 e 13) e naipe (1 - ouros, 2 - copas, 3 - paus, 4 - espadas) em sequência, e escreva: 1 - se as cartas contem uma canastra; 0 - se não contem. Considere como canastra apenas uma sequência crescente de 7 cartas do mesmo naipe de numeração contínua (cuja diferença entre duas cartas seja igual a 1).
Dica:Considere que podem ocorrer cartas de mesmo número e naipe (uso de mais de um baralho). Nesse caso a carta de mesmo número e naipe não é contada, mas não quebra a sequência.

58) Faça um programa que, baseando-se na sequência de fibonacci (0,1,1,2,3...), verifique se uma sequência de cinco números, faz parte da sequência, devolvendo como resposta "0" se faz parte, ou "1" se não faz parte. Dica:Considere que números "repetidos" fazem parte da sequência proposta, ou seja, devem ser analisados individualmente como os demais números fornecidos.
59) Faça um programa que leia um termo n que define o tamanho máximo que os lados de um triangulo retângulo podem assumir, sendo que os lados começam com tamanho 3, 4 e 5 (a,b,c). Em sequencia escreva todos os triângulos retângulos em que o valor máximo do lado é n e os três lados são inteiros. Cada triangulo deve ser escrito uma única vez em ordem crescente dos lados. Dica:O triangulo retângulo é dado como valido pela formula ($a^2 = b^2 + c^2$)

Vetores

- Um vetor é um conjunto de variáveis independentes, onde a cada elemento é atribuído um índice que permite referenciá-la individualmente.
- A forma de referenciar um elemento de um vetor na linguagem C é nome_do_vetor[índice_do_elemento].
- O índice do primeiro elemento de um vetor na linguagem C é 0.
- Um elemento de um vetor e uma variável, e pode ser utilizado em qualquer situação que uma variável simples, especificando o nome do vetor e o índice do elemento a ser referenciado.

Ex. Considere um vetor V de 5 elementos do tipo int (V[0], V[1], V[2], V[3], V[4]):

```
V[0]=0;
V[1]=V[0]+3;
printf("%d\n",V[2]);
```

- Na declaração de um vetor deve ser especificado o tipo de cada elemento e o número de elementos:
- Ex: int V[5]; /* vetor de 5 elementos do tipo int, de índices 0 a 4 */
- Obs: Na declaração especifica-se o número de elementos, assim, como o índice do primeiro elemento é 0, o índice do último elemento do vetor é de um a menos do que o tamanho declarado.
- Pode-se inicializar os elementos de um vetor já na declaração:
- int V[5]={1,2,3,4,5}; // declara um vetor de 5 elementos (com índices de 0 a 4) do tipo int, inicializados com os valores 1,2,3,4,5.

Leitura e escrita de um vetor

```
int v[10],i;
for (i=0;i<10;i++)
      scanf("%d",&v[i]);
for (i=0;i<10;i++)
      printf("%d ",v[i]);
```

Ordenação de um vetor pelo método da bolha (Bubble sort)

```
int v[10],i,j,aux;
for (i=0;i<9;i++)
      for (j=0;j<9;j++)
            if (v[j]>v[j+1]) {
                  aux=v[j];
                  v[j]=v[j+1];
                  v[j+1]=aux;}
```

Exercícios de Vetores

- 1) Escrever um programa que lê um vetor V[10] de inteiros e escreve os valores na ordem contrária à que foram digitados.
- 2) Escreva um programa que lê 10 valores inteiros. Escreva a seguir todos os valores pares e, após, escreva todos os valores ímpares.
- 3) Escreva um programa que leia 10 valores reais, calcule sua média aritmética e escreva, a seguir, os valores que são maiores que a média calculada.
- 4) Escreva um programa que leia um vetor V[10] de inteiros. Leia, a seguir, 5 valores e, para cada um, verifique se ele se encontra ou não no vetor, escrevendo uma mensagem apropriada.

- 5) Escrever um programa que leia um vetor V[10] e verifique se há valores repetidos, escrevendo ao final uma mensagem apropriada.
- 6) Escrever um programa que lê um vetor V[10] de inteiros e um vetor W[10] de inteiros e escreva os elementos que aparecem somente em V.
- 7) Escrever um programa que lê um vetor V[10] de inteiros e um vetor W[10] de inteiros e escreva os elementos que aparecem em V e W (interseção).
- 8) Escreva um programa que leia um vetor V[10] de inteiros e os escreva em ordem crescente.

- 9) Faça um programa que leia 10 datas de aniversário, cada uma formada por dia e mês, e as escreva em ordem cronológica crescente.
- 10) Faça um programa que leia, para 13 cartas de baralho, seu valor (inteiro entre 1 e 13) e naipe (inteiro entre 1 e 4) e identifique se as cartas contem uma canastra (sequencia de 7 cartas de numeração contínua do mesmo naipe). Considere que são utilizados dois baralhos.

- 11) Faça um programa que leia, para 20 candidatos a deputado, seu cadastro, partido (inteiro entre 1 e 10) e número de votos, e escreva o total de votos de cada partido. Os 3 dados correspondentes a cada candidato devem ser lidos juntos, antes de passar para o próximo candidato.
- 12) Faça um programa que leia, para 20 candidatos a deputado, seu cadastro, partido (inteiro entre 1 e 10) e número de votos, e escreva, para os 10 partidos, o número do partido e o total de votos, em ordem decrescente de votação. Os 3 dados correspondentes a cada candidato devem ser lidos juntos, antes de passar para o próximo candidato.

- 13) Faça um programa que leia, para 20 candidatos a deputado, seu cadastro, partido (inteiro entre 1 e 10) e número de votos, e escreva o cadastro, partido e número de votos do candidato mais votado de cada partido, em ordem crescente de partido.
- 14) Faça um programa que leia, para 20 candidatos a deputado, seu cadastro, partido (inteiro entre 1 e 10) e número de votos, e escreva o cadastro, partido e número de votos do candidato mais votado de cada partido, em ordem decrescente de votação.

- 15) Faça um programa que leia, para 20 candidatos a deputado, seu cadastro, partido (inteiro entre 1 e 10) e número de votos, e escreva o cadastro, partido e número de votos de todos os candidatos, em ordem crescente de partido e, para o mesmo partido, em ordem crescente de número de votos.
- 16) Faça um programa que leia 20 números reais. Escreva a quantidade mínima de elementos cuja soma seja maior ou igual à metade da soma total. Dica: Classifique o vetor e some os primeiros (ou últimos) elementos até atingir a metade da soma total.

- 17) Faça um programa que leia 10 datas, cada uma composta por dia, mês e ano, e escreva as 10 datas em ordem cronológica crescente.

- 18) Faça um programa que para 10 cartas de baralho leia o seu valor (entre 1 e 13) e naipe (1 - ouros, 2 - copas, 3 - paus, 4 - espadas), e ordene-as em ordem crescente de naipe e, para cada naipe, em ordem crescente de valor. Escreva a lista de cartas ordenada.
- 19) Faça um programa que para 9 cartas de baralho leia o seu valor (entre 1 e 13). O programa deve escrever quantas trincas a "mão" contém. Uma trinca é composta de três cartas com o mesmo valor. O naipe das cartas não importa. Cada carta pode aparecer no máximo em uma trinca.

- 20) Faça um programa que leia os 5 números obtidos em um arremesso de 5 dados no jogo de General, e verifique se os números contém um full-hand (3 números iguais, e os outros dois números iguais entre si, mas diferentes dos 3 primeiros. Ex. 3,3,3,2,2), e escreva: 1 - se os números contem um full-hand; 0 - se não contem.
- 21) Faça um programa que leia um vetor V(10) e um vetor W(3), e verifique e escreva a primeira posição do vetor V a partir da qual são encontrados, de forma contígua, os três elementos do vetor W. Caso o vetor W não ocorra em V, o programa deve escrever -1.

- 22) Faça um programa que leia um vetor de inteiros W[30]. Leia a seguir um vetor V[10] e verifique se a sequência de 10 números no vetor V ocorre no vetor W, escrevendo a posição do vetor W onde inicia a sequencia, caso ela exista, ou 0, caso a sequencia em V não exista em W.
- 23) Faça um programa que lê os conceitos de uma turma de N alunos, notas inteiras, de 0 a 10, até que seja digitado um valor ¡ 0 (que deve ser descartado) e escreva, para os valores de 0 a 10, o valor e o número de ocorrências do mesmo.

- 24) Crie um programa que lê um vetor G(13) representando o resultado de um teste de loteria esportiva, contendo os valores 1 (coluna 2), 2 (coluna do meio) e 4 (coluna 1). A seguir, para cada apostador, o programa lê o n° de seu cartão e suas apostas para cada um dos 13 jogos, sendo válidos os seguintes valores: 1 (coluna 2), 2 (coluna do meio), 3 (coluna 2 e do meio), 4 (coluna 1), 5 (coluna 1 e 2), 6 (coluna 1 e do meio) e 7 (triplo). O programa calcula, para cada apostador, o n° de acertos e escreve o n° do apostador e seu número de acertos. O programa para de ler quando o número do apostador for 0.

- 25) Faça um programa que leia valores até que seja digitado um valor negativo e escreva os 10 últimos valores positivos digitados. Considere que serão digitados pelo menos 10 valores positivos. O programa deve escrever os valores na mesma ordem em que foram digitados pelo usuário.
- 26) Crie um programa que lê o número de matrícula de cada aluno e suas respectivas notas em três provas, em uma turma de 10 alunos. Após, calcular a média harmônica (MH) de cada aluno e verificar o conceito obtido (se MH<6.0, conceito é 0. Se MH<7.0, conceito é 1. Se MH<8.0 o conceito é 2. Se MH<9.0 o conceito é 3. Se MH>=9.0, conceito é 4). Escrever seu número de matrícula e seu conceito. Dica:Lembre que se uma das três notas é 0, a média harmônica é 0.

- 27) Na apuração de um processo de eleição as vagas disponíveis são distribuídos entre os partidos proporcionalmente aos votos obtidos pelo partido. As sobras são distribuídas da seguinte forma: enquanto houver sobras, calcula-se para cada partido a razão entre o número de votos e o número de cadeiras já obtidas + 1. Uma das sobras é atribuída ao partido de maior razão. Isso se repete até que não haja mais sobras. Faça um programa que leia, para 10 partidos, de 1 a 10, o seu número de votos. Leia, a seguir, o total de vagas e escreva, ao final, para os partidos de 1 a 10, o número do partido e o número de cadeiras obtidas.

- 28) Escreva um programa que lê um vetor V[10] e escreva, para cada valor diferente que aparece no vetor, o valor e o número de vezes que o mesmo ocorre no vetor. Escreva os valores em ordem crescente de valor.
- 29) Faça um programa que leia um vetor V[10] e ao final escreva o elemento que aparece mais vezes neste vetor.
- 30) Faça um programa que leia um vetor V[15], já ordenado. Leia, a seguir, um valor N e verifique se ele se encontra no vetor escrevendo a posição em que se encontra. Caso o valor não esteja no vetor, escreva -1. Tente escrever um programa da forma mais eficiente possível.

- 31) Faça um programa que leia os 10 primeiros valores de um vetor V[20], valores estes já ordenados em ordem crescente. Leia, a seguir, 10 valores e insira cada um deles na posição correta do vetor V, de modo a mantê-lo ordenado, deslocando em uma posição para a direita todos os valores que vêm após ele. Escreva, ao final, o vetor V (ordenação por inserção).
- 32) Na ordenação por contagem, conta-se, para cada elemento do vetor, o número de elementos menores ou iguais a ele. O valor da contagem é a posição que o elemento ocupará no vetor ordenado. Faça um programa que leia um vetor V[20] e ordene-o em ordem crescente, escrevendo o vetor ordenado. Considere que não há repetição de valores no vetor.

- 33) Uma loja decidiu fazer um sorteio de prêmios para seus clientes. Serão sorteados 10 números, porém, a loja não quer que um mesmo número receba mais do que um prêmio. Faça um programa que leia cada número sorteado até que tenham sido sorteados 10 números diferentes. O programa escreve os 10 números (distintos) sorteados.
- 34) Crie um programa que leia um vetor v[10]. Leia a seguir um valor N e faça, no vetor V, uma rotação circular para a esquerda (rotate left) de N posições. Na rotação circular para a esquerda, cada elemento é copiado para a posição à esquerda, e o primeiro elemento é copiado para a última posição. Escreva, ao final, o vetor alterado.

- 35) Faça um programa que leia um número inteiro e escreva os dígitos binários correspondentes a esse número (máximo 64 dígitos binários). Dica:Para converter um número decimal em binário pode-se proceder sucessivas divisões por dois até que o quociente seja igual a zero. O número binário é obtido considerandos restos obtidos nas divisões, mas em ordem contrária.
- 36) Leia um vetor de 10 posições que contem números positivos e negativos. Em seguida, descubra qual a sequencia de valores dentro dele (intervalo de valores contíguos no vetor) que resulta na maior soma, escrevendo ao final a soma encontrada.

- 37) A sequencia "Look & Say" funciona da seguinte forma: O seu primeiro termo é sempre "1", e a definição do próximo termo a partir do anterior é "O n° de vezes que um número de repete até que a sequencia seja quebrada""O número que estava repetindo"...

  Veja do seguinte modo:

  1° Termo: 1 (Termo inicial em "1")

  2° Termo: 11 (Antes tínhamos "1" vez o número "1")

  3° Termo: 21 (Antes tínhamos "2" vezes o número "1")

  4° Termo: 1211 (Antes tínhamos "1" vez o número "2" e "1" vezes o número "1")

  5° Termo: 111221

  6° Termo: 312211

  ...

- Elabore um programa que calcule o enésimo termo dessa sequência. Considere que o número é muito grande e você precisará de 1000 casas no vetor que irá aglomerá-lo.

Variáveis do tipo caracter (char)

- São variáveis que podem conter um caracter, podendo ser um uma letra, dígito ou caracter especial. Os caracteres são armazenados utilizando uma representação chamada ASCII (American Standard Code for Information Interchange), que representa cada caracter por um valor numérico entre 0 e 255.
- Caracteres de tabulação
  9 - Horizontal Tab - HT
  10 - Line Feed - LF
  13 - Carriage Return (Retorno de Carro) - CR

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

- Caracteres especiais

32 - Space - SPC		
33 - !	34 - ”	35 - #
36 - $	37 - %	38 - &
39 - ’	40 - (	41 -)
42 - *	43 - +	44 - ,
45 - -	46 - .	47 - /

- 48 a 57 - Dígitos de 0 a 9
- 65 a 90 - Letras maiúsculas de A a Z
- 97 a 122 - Letras minúsculas de a a z

- Variáveis do tipo caracter são declaradas com o tipo char.
  - Ex: char a,b,c;
- Constantes do tipo char são representadas pelo caracter entre aspas simples. Ex: let='T'; (a variável let recebe o caracter 'T')
- O especificador de formato para variáveis char é %c.
- O programa a seguir escreve a posição na tabela ASCII e o caracter das posições 32 a 126.

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

int main() {

int i;
for ( i = 32; i < 127; i++ ) {
printf( "%c[%d]\n", i , i );
}
system("pause");
return 0;
}
```

Função getch()

- Variáveis char podem ser lidas com a função getch( ), da biblioteca conio.h.
- Ex:

  c=getch( ); /* fica esperando o usuário digitar um caracter e quando for digitado o caracter é colocado na variável c. */
- Faça um programa que fique lendo teclas até que seja digitada a tecla 'f' e, quando forem tecladas as teclas das setas, escreva uma mensagem correspondente à tecla teclada. Ex: "Tecla para cima", "Tecla para a esquerda".
- Teclas diferentes das setas e de 'f' devem ser ignoradas.

Strings

- São vetores de caracteres utilizado para manipulação de textos em C.
- Declaração: char nome[10]; // vetor de 10 posições do tipo caracter.
- Obs: Uma das posições do vetor é utilizada para armazenar o delimitador de fim de string (valor inteiro 0, representado por '\0').
- Assim, para armazenar N caracteres é necessário declarar um vetor com N+1 posições.

- Leitura de um string:

  scanf("%s",nome);
  - Obs: Como um string é um vetor, não é necessário o & para a leitura do mesmo.
- Escrita de um string:

  printf("%s",nome);

1) Faça um programa que leia um string e o escreva.

```
#include <stdio.h>
int main()
{
char nome[10];
scanf("%s",nome);
printf("%s",nome);
}
```

Obs: O scanf utiliza o espaço em branco como delimitador de fim de string na leitura. Pode-se ler strings com o comando gets(string), que lê todos os caracteres até ser digitado ENTER.

Funçoes para manipulação de strings (colocar no início #include < *string.h* >)

- **strcmp**(str1,str2) - compara dois strings em ordem lexicográfica e retorna:
  - 0 - se os strings são iguais
  - $\leq$0 - se str1 é menor que str2 (considerando ordem lexicográfica)
  - $\geq$0 - se str1 é maior que str2 (considerando ordem lexicográfica)

ex:

```
if (strcmp(nome1,nome2)<0)
    printf("%s",nome1)
else printf("%s",nome2);
```

- 2) Faça um programa que leia dois nomes e escreva-os em ordem alfabética.
- Pode-se declarar uma lista de strings como uma matriz de caracteres:
  - char nomes[20][15]; // declara uma lista de 20 strings, cada um com até 14 caracteres (e mais um espaço para o delimitador de fim de string)
- Pode-se inicializar um string na declaração:

  char nome[20]="João";
- também pode-se inicializar uma lista de strings:

  char nomes[5][15]={"João","Maria","Zefa","Juca","Jeca"};
  char unidades[10][10]={"zero","um","dois","tres","quatro",
  "cinco","seis","sete","oito","nove"};

- **strcpy**(destino, origem) - copia o string de origem para o string de destino.

  Ex:

  ```
  char nome[20];
  strcpy(nome,"João"); // copia "João" para o string nome
  ```

- **strlen**(str) - retorna o tamanho do string em str.

  Ex:

  ```
  a=strlen("João"); // a variável "a" recebe o valor 4
  ```

- **strcat**(str1,str2) - concatena uma cópia de str2 no final de str1.

  Ex:

  ```
  strcpy(st1,"João");
  strcat(st1," da Silva");//st1 ficará com "João da Silva"
  ```

- 1.Fazer um programa que leia uma palavra e diga se ela é ou não um palíndromo. Ex: ARARA, ovo.
- 2.Idem ao anterior, mas sem distinção de maiúsculas e minúsculas. Ex: Arara, Ovo, OsSo.
  - Para converter um char de minúscula para maiúscula, utilize a função
    char toupper(char letra)
  - que recebe um caracter e retorna o caracter maiúsculo equivalente
- 3.Fazer um programa que leia uma letra (L) e um número (N), a seguir gere uma string contendo N letras L.
- 4.Fazer um programa que leia um número inteiro positivo, converta-o em uma string e escreva-a.
- 5.Fazer um programa que leia um número inteiro positivo, converta-o em hexadecimal, armazenando o resultado em uma string e escreva-a.

- 6.Fazer um programa que leia uma string até que seja um número hexadecimal válido, após converta esse número lido para decimal, armazenando-o em uma variável inteira e escreva-a.
- 7.Fazer um programa que leia uma string e a partir desta gere uma nova duplicando cada caracter da string original. Escreva a nova string. Ex: "OI" => "OOII"; "PROVA 1" => "PPRROOVVAA 11"
- 8.Fazer um programa que leia uma string e a partir desta gere uma nova contendo um espaço em branco entre cada caracter da string original. Escreva a nova string. Ex: "LIVRO DE C" => "L I V R O D E C"

- 9. Faça um programa em C que inverta os caracteres de uma string. Por exemplo, se a string for "UNIVASF", deve ser convertida a "FSAVINU".
- 10. Faça um programa em C que leia uma string s, um caracter chAtual, um caracter chNovo e substitua todo caracter chAtual de s pelo chNovo. O programa deve retornar também o número de substituições.
- 11. Faça um programa em C que leia uma string s, um caracter ch, um inteiro pos e insira o caracter ch na posição pos da string s.
- 12. Faça um programa em C que leia uma string s1, uma string s2, um inteiro pos e insira a string s2 em s1 na posição pos.

- 13. Faça um programa em C que leia uma string s1, uma string s2, um inteiro n e copie os n primeiros caracteres da string s1 na string s2.
- 14. Faça um programa em C que leia uma string s1, uma string s2, um inteiro n e copie os n últimos caracteres da string s1 na string s2.
- 15. Faça um programa em C que leia uma string s1, uma string s2, um inteiro n, um inteiro inicio e copie os n caracteres a partir da posição inicio da string s1 na string s2.

- 16.Fazer um algoritmo que leia o nome completo de uma pessoa, separando-o em partes. Considerar como delimitador(es) espaço(s) em branco. Escreva cada parte do nome em uma nova linha. Exemplo:

Lê	Escreve
Paulo Henrique Silveira	Paulo Henrique Silveira
Pedro Silva	Pedro Silva

- 17.Fazer um algoritmo que leia o nome de uma pessoa e após gere o seu email (utilizando letras minúsculas) considerando a primeira letra dos primeiros nomes completando com as letras do último nome até, no máximo, oito caracteres. Exemplos:

Nome da Pessoa	Email
Paulo Henrique Silveira	phsilvei
Pedro Silva	psilva

- 18. Idem ao exercício anterior, mas armazene os emails gerados em um vetor evitando geração de emails duplos, trocando a(s) última(s) letras do mesmo por números sequenciais a partir de um. Exemplos:

Nome da Pessoa	Email
Paula Silva	psilva
Paulo Renato Henrique Miguel Silveira	prhmsilv
Plínio Roberto Humberto Moraes Silva	prhmsil1
Pedro Silva	psilva1
Patricia Silva	psilva2

- 19. Fazer um algoritmo que leia nomes e salários de dez funcionários e após escreva somente os nomes em ordem decrescente de salário e finalmente escreva os nomes em ordem alfabética e os respectivos salários.
- 20. Fazer um algoritmo que leia uma string contendo uma data no formato DD/MM/AAAA. Separe o dia, o mês e o ano, armazenando-os em três variáveis inteiras. Caso o formato não seja esse, armazene -1 nessas variáveis e finalmente escreva essas três variáveis.

- 21. Faça um programa que leia um número inteiro entre 0 e 99 e gere um string com a representação do número por extenso, e escreva o string ao final.

Comando de seleção múltipla

- O comando switch-case é um comando de seleção de uma dentre diversas alternativas. Seu formato é:

```
switch (expressão) {
        case valor_1: sequencia de comandos;
                      break;
        case valor_2: sequencia de comandos;
                      break;
        default: sequencia de comandos
}
```

- O valor da expressão é comparado com o valor de cada constante especificada no case. Quando um valor igual for encontrado, a sequência de comandos associados é executada, até encontrar o comando break.
- Se a lista de comandos de uma das entradas não terminar por break, a lista de comandos da entrada seguinte será executada.
- Se a expressão não coincidir com nenhum valor, a sequencia de comandos associada ao *default* é executada. Caso não haja uma entrada *default*, nenhum comando será executado.

Ex:

```
switch (dia) {
case 1: printf("Domingo\n");break;
case 2: printf("Segunda-feira\n");break;
case 3: printf("Terça-feira\n");break;
case 4: printf("Quarta-feira\n");break;
case 5: printf("Quinta-feira\n");break;
case 6: printf("Sexta-feira\n");break;
case 7: printf("Sábado-feira\n");break;
default:printf("Dia␣inválido\n");
}
```

- Exercício: Faça um programa que leia um valor inteiro entre 0 e 999 e escreva o valor por extenso.

Matrizes

- Matrizes são conjuntos de variáveis organizados em uma estrutura de duas dimensões (i.e. linhas e colunas)
- Da mesma forma que vetores, para referenciar um elemento de uma matriz deve-se especificar o número da linha e da coluna que se quer referenciar
- Ao declarar uma matriz deve-se especificar o número de linhas e colunas da mesma, bem como o tipo de valores com que ela trabalha (float, int...):
  - Ex: int Mat[5][5];

	0	1	2	3	4
0	1	0	0	0	0
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	0	0	0	0	1

Matrizes

- Matrizes podem ter mais de duas dimensões, mas nesse semestre trabalharemos apenas com matrizes bidimensionais.
- Assim como nos vetores, o número da primeira linha é 0, e o número da primeira coluna de cada linha é 0.
- A forma de referenciar um elemento de uma matriz na linguagem C é nome_da_matriz[número da linha][número da coluna].
- Um elemento de uma matriz é uma variável, e pode ser utilizado em qualquer situação que uma variável simples, especificando o nome da matriz e a linha e coluna do elemento a ser referenciado.

Operações sobre matrizes

- Assim como vetores, operações sobre matrizes são feitas elemento a elemento
- Normalmente constituem de dois **for** encadeados, para gerar todas as combinações de linhas e colunas. Ex:

```
for (i=0;i<3;i++){
      for (j=0; j<3;j++){
            M[i][j]=0;
      }
}
```

i	j	
0	0	M[0,0]←0
0	1	M[0,1]←0
0	2	M[0,2]←0
1	0	M[1,0]←0
1	1	M[1,1]←0
1	2	M[1,2]←0
2	0	M[2,0]←0
2	1	M[2,1]←0
2	2	M[2,2]→0

- A figura abaixo ilustra os índices (linha e coluna) dos elementos de uma matriz 5x5.
- M00000100 - Faça um programa leia uma matriz M[5][5]e calcule e escreva a soma dos elementos da diagonal principal (em negrito na figura abaixo).

	0	1	2	3	4
0	**0,0**	0,1	0,2	0,3	0,4
1	1,0	**1,1**	1,2	1,3	1,4
2	2,0	2,1	**2,2**	2,3	2,4
3	3,0	3,1	3,2	**3,3**	3,4
4	4,0	4,1	4,2	4,3	**4,4**

- A figura abaixo ilustra os índices (linha e coluna) dos elementos de uma matriz 5x5.
- M00000150 - Faça um programa que leia uma matriz M[5][5] e calcule e escreva a soma dos elementos da diagonal secundária (em negrito na figura abaixo).

	0	1	2	3	4
0	0,0	0,1	0,2	0,3	**0,4**
1	1,0	1,1	1,2	**1,3**	1,4
2	2,0	2,1	**2,2**	2,3	2,4
3	3,0	**3,1**	3,2	3,3	3,4
4	**4,0**	4,1	4,2	4,3	4,4

- Pode-se identificar a posição de um elemento em relação à diagonal principal (acima, abaixo ou exatamente na diagonal principal) pela relação entre seus índices (linha e coluna)
  - Elementos **exatamente** na diagonal principal: linha $=$ coluna
  - Elementos **acima** da diagonal principal: linha $<$ coluna
  - Elementos **abaixo** da diagonal principal: linha $>$ coluna

	0	1	2	3	4
0	**0,0**	0,1	0,2	0,3	0,4
1	1,0	**1,1**	1,2	1,3	1,4
2	2,0	2,1	**2,2**	2,3	2,4
3	3,0	3,1	3,2	**3,3**	3,4
4	4,0	4,1	4,2	4,3	**4,4**

- De forma semelhante pode-se identificar a posição de um elemento em relação à diagonal secundária (acima, abaixo ou exatamente na diagonal principal) pela soma de seus índices (linha e coluna). Em uma matriz NxN:
  - Elementos **exatamente** na D.S.: linha + coluna = N - 1
  - Elementos **acima** da D.S.: linha + coluna < N - 1
  - Elementos **abaixo** da D.S: linha + coluna > N - 1

	0	1	2	3	4
0	0,0	0,1	0,2	0,3	**0,4**
1	1,0	1,1	1,2	**1,3**	1,4
2	2,0	2,1	**2,2**	2,3	2,4
3	3,0	**3,1**	3,2	3,3	3,4
4	**4,0**	4,1	4,2	4,3	4,4

- Na declaração de uma matriz devem ser especificados o tipo de cada elemento, o número de linhas e o número de colunas.
- Ex:

  int m[5][5]; /* matriz de 25 elementos do tipo int, de linhas 0 a 4, cada linha tendo as colunas 0 a 4 */
- Pode-se inicializar os elementos de uma matriz já na declaração:

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

  // declara uma matriz de 3 linhas (de 0 a 2) e 3 colunas (de 0 a 2) do tipo int, já inicializadas.

1) Faça um programa que leia uma matriz m[3][3]. Calcule e escreva, após, a soma de todos os elementos da matriz.

- A figura abaixo ilustra os índices (linha e coluna) dos elementos de uma matriz 5x5.

	0	1	2	3	4
0	**0,0**	0,1	0,2	0,3	0,4
1	1,0	**1,1**	1,2	1,3	1,4
2	2,0	2,1	**2,2**	2,3	2,4
3	3,0	3,1	3,2	**3,3**	3,4
4	4,0	4,1	4,2	4,3	**4,4**

2) Faça um programa que leia uma matriz m[4][4] e calcule e escreva:

a) A soma dos elementos da diagonal principal.

b) A soma dos elementos da diagonal secundária.

c) A soma dos elementos ACIMA da diagonal principal.

d) A soma dos elementos ACIMA da diagonal secundária.

e) A soma dos elementos ACIMA da diagonal principal e ABAIXO da secundária

3) Faça um programa que leia uma matriz m[4][4] e:

a) Troque todos os elementos da linha 1 pelo elemento correspondente na linha 3.

b) Troque cada elemento da coluna 0 pelo elemento correspondente na coluna 2.

Escreva ao final a matriz alterada.

- 4) Faça um programa que leia uma matriz M[5][5], possivelmente com elementos repetidos. Leia, a seguir, 5 valores e, para cada um, verifique se o valor ocorre ou não na matriz, escrevendo a posição (linha e coluna) em que foi encontrada a primeira ocorrência do mesmo e, caso ele não exista na matriz, a mensagem "Não tem".
- 5) Faça um programa que leia uma matriz m[6][5]. Escreva, ao final, a soma dos elementos de cada linha e a soma dos elementos de cada coluna.
- 6) Faça um programa que leia uma matriz M[5][5] e escreva o maior valor existente na matriz, bem como a linha e coluna onde o valor ocorre.

Exercícios de matrizes em geral:

- 1) Escreva um programa que leia duas matrizes numéricas A[3][4] e B[3][4] e gere e escreva uma matriz inteira C[3][4], tal que um elemento C[i][j]= 1 se os elementos nas mesmas posições das matrizes A e B forem iguais, e 0 em caso contrário. O programa deve, ao final, escrever a matriz C.
- 2) Uma matriz identidade é uma matriz que possui 1 em todos os elementos da diagonal principal, e 0 em todas as outras posições. Faça um programa que leia uma matriz M[5][5] e verifique se é uma matriz identidade escrevendo uma mensagem apropriada.

- 3) Faça um programa que leia uma matriz M[5][5] e escreva o número da linha que contenha a maior soma de seus elementos. Considere que a matriz só contem valores positivos.
- 4) Faça um programa que leia uma matriz M[5][5] e escreva o número da linha que contenha a maior soma de seus elementos. Considere que a matriz pode conter valores positivos e negativos.

5) Faça um programa que leia uma matriz M[5][5] e gere dois vetores SomaLin[5] e SomaCol[5], com a soma dos elementos de cada linha e a soma dos elementos de cada coluna da matriz M. Escreva ao final os vetores Somalin e Somacol.

- 6) Uma matriz é dita Diagonalmente Dominante se atende às duas condições abaixo::

  a) em todas as linhas o elemento da diagonal principal é maior ou igual à soma dos outros elementos da linha e,
  b)há pelo menos uma linha em que o elemento da diagonal principal é MAIOR que a soma dos outros elementos da linha (não basta que seja igual).

  Faça um programa que leia uma matriz M[4][4] e verifique se é diagonalmente dominante escrevendo:

  1 - Se é diagonalmente dominante;
  0 - Se não é diagonalmente dominante

- 7) Faça um programa que leia uma matriz M[5][5], onde cada posição contem um número entre 0 e 9 e cada linha da matriz representa um número de 5 dígitos. O programa deve calcular a soma dos 5 números contidos na matriz colocando o resultado em um vetor Soma[6]. Escreva ao final o vetor Soma.
- 8) Faça um programa que leia uma matriz M[5][5], onde cada posição contem um número entre 0 e 9 e cada linha da matriz representa um número de 5 dígitos. O programa deve encontrar a linha que contem o maior dos 5 números e escrever o número.

- 9) Faça um programa que leia uma matriz M[5][5], onde cada posição contem um número entre 0 e 9 e cada linha da matriz representa um número de 5 dígitos. O programa deve ordenar os 5 números em ordem crescente, e escrever a matriz com os números ordenados.
- 10) Faça um programa que leia dois valores D e DS, correspondendo a um dia do mês e um dia da semana (1-domingo, 2-segunda,... 7-sábado) e preencha uma matriz Folhinha[6][7] com o calendário correspondente ao mês do dia digitado. A primeira coluna da matriz contem os domingos, a segunda coluna contem as segundas e assim por diante. O programa deve escrever a matriz gerada. As posições não utilizadas da matriz devem conter 0's.

- 11) Faça um programa que leia uma matriz M[5][5] e um valor N e multiplica cada valor da matriz M por N e coloca o resultado em um vetor V[25]. Escreva ao final o vetor V.
- 12) Faça um programa que leia uma matriz M[5][6] e divide todos os 6 elementos de cada linha pelo valor do menor elemento EM MÓDULO da linha. Escrever a matriz modificada.
- 13) Faça um programa que leia uma matriz M[5][5], possivelmente com elementos repetidos. Leia, a seguir, 5 valores e, para cada um, verifique se o valor ocorre ou não na matriz, escrevendo a posição (linha e coluna) em que foi encontrada a primeira ocorrência do mesmo e, caso ele não exista na matriz, a mensagem "Não tem".

- 14) Uma matriz é dita Matriz de Toeplitz se, em cada diagonal paralela à diagonal principal, todos os elementos são iguais. Assim, um exemplo de Matriz de Toeplitz é:

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

Faça um programa que leia uma matriz M[1..5,1..5] e verifique se é uma Matriz de Toeplitz, escrevendo:

1 - Se é uma matriz de Toeplitz;

0 - Se não é uma matriz de Toeplitz

- 15) - Uma matriz é dita Circulante cada elemento é igual ao elemento imediatamente acima à esquerda, e se o primeiro elemento de cada coluna é igual ao último elemento da coluna anterior. Assim, um exemplo de Matriz Circulante é:

1	9	7	8	2
2	1	9	7	8
8	2	1	9	7
7	8	2	1	9
9	7	8	2	1

Faça um programa que leia uma matriz M[1..5,1..5] e verifique se é uma Matriz Circulante, escrevendo:

1 - Se é uma matriz Circulante;

0 - Se não é uma matriz Circulante

- 16) Faça um programa que leia uma matriz M[1..3,1..3] e para cada linha divida toda a linha pelo seu elemento da primeira coluna. Após isso, a partir da segunda linha, de cada linha subtraia toda a a primeira linha elemento a elemento.
  ex. a matriz:

2	4	6
3	9	6
2	8	4

1	2	3
1	3	2
1	4	2

1	2	3
0	1	-1
0	2	-1

  Escreva a matriz alterada.

- 17) Um quadrado mágico de ordem N (sendo N um número ímpar) é um arranjo de número de 1 a NxN em uma matriz quadrada de tal modo que a soma de cada linha, coluna e diagonal é a mesma. A matriz abaixo representa um quadrado mágico de ordem 5.

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

- A regra para gerá-lo é relativamente fácil de observar:
  - Comece com 1 no meio da primeira linha.
  - Siga para cima e para a esquerda diagonalmente (quando sair do quadrado suponha que os lados superior e inferior estão unidos ou que os lados da direita e da esquerda estão unidos, conforme for o caso).
  - Em cada quadrado que passar coloque o valor do quadrado anterior mais 1 (um).
  - Quando a próxima casa estiver já preenchida, desça um quadrado e continue seguindo a diagonal até ter preenchido todos os quadrados.
- Faça um programa que lê um número N ímpar, menor ou igual a 99, e gere e escreva, para o número lido, o seu quadrado mágico.

18) Faça um programa que leia duas matrizes M[3][3] e N[3][3] (primeiro a matriz M, e após a matriz N) e calcule o produto matricial MxN colocando o resultado em uma matriz P[3][3]. Escreve ao final a matriz P.

19) Uma matriz de adjacências, para um mapa rodoviário, é composta de elementos 0's e 1's, sendo que M[i][j]= 1 se existe uma ligação rodoviária da cidade i para a cidade j e M[i][j]= 0 em caso contrário. Essa matriz será simétrica se para todo caminho i,j existir o caminho j,i (estradas de mão dupla). Uma característica dessa matriz é que na matriz M^2, obtida pelo produto matricial de M por M, cada posição M[i][j] contém o número de caminhos entre as cidades i e j que passam exatamente por 1 outra cidade. E M^3 contem a quantidade de caminhos que passam por 2 outras cidades. E assim por diante. Escreva um programa que leia uma matriz de adjacências M para um conjunto de 5 cidades e, após, leia os valores c1 e c2 (ambos menores ou iguais a 5) representando duas cidades quaisquer do conjunto. O programa deve descrever a quantidade de caminhos entre c1 e c2 que passam exatamente por 1 outra cidade.

20) As pirâmides têm a base quadrada, sendo que a única forma de se atingir o topo é seguir em espiral pela borda. Escreva um programa que leia um valor N e, a seguir, uma matriz quadrada A[n][n] de ordem no máximo igual a 10, e verifique se a matriz é equivalente a uma pirâmide inca, ou seja, se partindo-se do canto superior esquerdo da matriz, no sentido horário, em espiral, a posição seguinte na ordem é o inteiro consecutivo da posição anterior. O programa deve escrever:

1 - Se a matriz forma uma pirâmide inca;

0 - Em caso contrário.

21) Faça um programa que leia uma matriz M[3][20], onde cada posição contem um número entre 0 e 9 e cada linha da matriz representa um número de 20 dígitos. O programa deve calcular a soma dos 3 números contidos na matriz colocando o resultado em um vetor Soma[21]. Escreva ao final o vetor Soma. Utilize o programa para calcular a soma dos números: 12345678901234567890, 93579135799357913579 e 99999999999999999999

22) Faça um programa que leia uma matriz M[5][5], onde cada posição contem um número entre 0 e 9 e cada linha da matriz representa um número de 5 dígitos. O programa deve encontrar a linha que contem o maior dos 5 números e escrever o número.
23) Faça um programa que leia uma matriz M[5][5], onde cada posição contem um número entre 0 e 9 e cada linha da matriz representa um número de 5 dígitos. O programa deve ordenar os 5 números em ordem crescente, e escrever a matriz com os números ordenados.

24) Faça um programa que leia uma matriz M[5][5] e um valor N e multiplica cada valor da matriz M por N e coloca o resultado em um vetor V[25]. Escreva ao final o vetor V.
25) Faça um programa que leia uma matriz M[5][6] e divide todos os 6 elementos de cada linha pelo valor do menor elemento EM MÓDULO da linha. Escrever a matriz modificada.

- 26) Uma matriz é dita triangular superior se todos os elementos abaixo da diagonal principal são iguais a zero, e há pelo menos um elemento nulo acima da diagonal principal.

  Da mesma forma, uma matriz é dita triangular inferior se todos os elementos acima da diagonal principal são iguais a zero, e há pelo menos um elemento não nulo abaixo da diagonal principal.

  E uma matriz é dita diagonal se os elementos não nulos ocorrem somente na diagonal principal.

  Faça um programa que leia uma matriz M[1..5,1..5] e escreva:

  0 - Se a matriz é uma matriz diagonal;
  1 - Se é triangular superior;
  2 - Se é triangular inferior;
  3 - Se não é nenhuma das anteriores

- 27) Escrever um programa que lê uma matriz M[1..3,1..3], contendo uma posição de jogo da velha, com valores 0 (casa livre), 1 (marcada com cruzinha) ou 2 (marcada com bolinha) e escreva:

  1 - se o jogador 1 venceu o jogo (alguma linha, coluna ou diagonal com o mesmo valor);
  2 - se o jogador 2 venceu o jogo;
  3 - se o jogo terminou empatado (não há mais lances e ninguém ganhou);
  4 - se o jogo ainda não terminou (há lances por jogar e ninguém ainda venceu)

28) Na Teoria de Sistemas, define-se como elemento minimax de uma matriz o menor elemento da linha em que se encontra o maior elemento da matriz. Escreva um programa que leia uma matriz A[5][5] e determine o seu elemento minimax. O programa deve, ao final, escrever o valor do elemento minimax, bem como a linha e coluna onde ocorreu.

- Faça um programa para ler um número de 1 a 999 e escrevê-lo por extenso, utilizando o comando switch.
- Refaça o exercício anterior os switch por uma lista de strings.

Matrizes de Caracteres

- Matrizes de caracteres (p.ex. char Nomes[10][20]) podem ser tratadas como listas de strings, onde cada linha da matriz contem um string.
- Todas as funções aplicadas a strings (strcmp, strcpy, strcat. strlen...) podem ser aplicadas a linhas da matriz.
- Da mesma forma, a matriz pode ser inicializada com uma lista de nomes, onde cada nome ocupará uma das linhas da matriz.
- char Nomes[5][20]={"João", "José","Juca","Jaca","Jeca"};

Para imprimir a lista:

```
for (i=0;i<5;i++)
printf("%s\n",Nomes[i]);
```

Para copiar um string para uma posição da lista:

```
strcpy(Nomes[0],"Jájá");
```

Exercícios de listas de strings

1) Faça um programa para ler uma lista de 10 nomes e escrevê-los em ordem alfabética.

2) Faça um programa para ler duas listas de 5 nomes em cada lista, e escrever os nomes que aparecem em ambas as listas (interseção das listas).

3) Faça um programa para ler duas listas de 5 nomes em cada lista, e escrever os nomes que aparecem somente na primeira lista.

4) Faça um programa que leia 3 listas de 5 nomes cada, e escreva os nomes que estão nas duas primeiras listas (interseção) mas não estão na terceira lista.

5) Faça um programa que leia 3 listas de 5 nomes cada, e escreva os nomes que estão nas 3 listas (interseção).

6) Faça um programa que leia 2 listas de 5 nomes cada, e escreva os nomes que aparecem somente em uma das listas.

7) Faça um programa que leia uma lista de 10 nomes e escreva o nome que aparece mais vezes na lista e o nome que aparece menos vezes na lista.

8) Faça um programa que leia palavras e procure na matriz de caracteres abaixo. As palavras podem ocorrer na horizontal, vertical e diagonal, em qualquer orientação (de cima para baixo, baixo para cima, esquerda para a direita, direita para a esquerda...):

```
char mat[10][11]={"xanaeotina",
"xxnxnxvxxt",
"beaxioxaxu",
"cxxxrobnxr",
"xaxxexxgxi",
"xxtehaxexn",
"cxxirnxlxg",
"xaxxadhiix",
"xxtxcxnnnx",
"caihposaxx"};
```

```
char matriz[10][12] =       {{'A','B','C','D','E','F',
{'M','N','O','P','M','R','S','T','U','V','W','X'},
{'Y','Z','T','E','E','T','E','A','H','G','D','E'},
{'P','R','O','R','G','A','M','A','I','J','H','T'},
{'Q','W','O','M','A','I','R','O','G','L','A','F'},
{'D','F','G','H','U','K','C','V','B','N','O','P'},
{'D','L','O','A','M','A','R','G','O','R','P','I'},
{'F','I','B','O','N','A','C','C','I','E','R','T'},
{'A','G','U','A','I','F','P','R','I','M','O','S'},
{'A','B','C','D','L','F','G','H','I','J','H','I'}};
```

Funções rand( ) e srand ( )

- Para gerar valores randômicos utiliza-se a função rand( ), da biblioteca stdlib.h.
- Cada chamada da função rand( ) retorna um valor inteiro entre 0 e RAND_MAX (constante definida em stdlib).
- Para inicializar o gerador de números randômicos no início do programa chama-se a função srand (time(NULL));

Ex:

```
#include < stdio.h >
#include < stdlib.h >
#include < time.h >
int main ()
{
srand ( time(NULL) );
printf ("Um numero entre 0 e RAND_MAX (%d): %d\n", RAND_MAX, rand());
printf ("Um numero entre 0 e 99: %d\n", rand()%100);
printf ("Um numero entre 20 e 29: %d\n", rand()%10+20);
return 0;
}
```

1) Faça um programa que preencha uma matriz M[3][3] com valores randômicos entre 100 e 200 e escreva a matriz gerada. A seguir copie todos os elementos da matriz para um vetor, ordene-os em ordem decrescente e copie de volta para a matriz, escrevendo a matriz ordenada.

Funções

- Funções são trechos de código que podem ser chamados a partir de diversos pontos diferentes do programa.
- Ao invés de escrever um trecho de código diversas vezes, escreve-se o código apenas uma vez e ele é chamado diversas vezes.
- A linguagem C possui diversas funções prontas. Essas funções encontram-se em bibliotecas, declaradas nas cláusulas #include.
- Assim #include $< math.h >$ informa ao compilador que o programa utilizará funções matemáticas da biblioteca math.h.

- Além das funções pré-definidas na linguagem, o programador pode especificar suas próprias funções. Algumas vantagens do uso de funções são:
  - Reduzem o tamanho do programa, eliminando a repetição de código;
  - Melhoram a legibilidade do programa, "quebrando" um programa grande em partes menores, individualmente mais compreensíveis.
- Ao especificar uma função, o programador deve especificar que valores que ela irá receber, o tipo de valores que irá receber, e o tipo de valor que será retornado por ela (se for o caso). Os valores passados para uma função, que ela utilizará para efetuar os cálculos necessários, se chamam "parâmetros" ou "argumentos" da função.
- Os parâmetros enviados à função devem ser em mesma quantidade, tipo e ordem dos parâmetros definidos no cabeçalho da função.

- O formato geral de especificação de uma função é assim:

```
<tipo de valor de retorno> <nome da função> (<lista de parâmetros e tipo de cada um>)
    {
    <comandos>
    }
```

- Por exemplo, uma função que receba um valor inteiro e retorne o fatorial desse número pode ser escrita assim:

```
int fat (int n)
        {
        int i,f=1;
        for (i=1;i<=n;i++)
                f=f*i; // cálculo do fatorial
        return f; // especifica o valor que será retornado
        }
```

- E na chamada da função deve ser passado o parâmetro esperado.
- No caso da função fat, deve ser passado o valor de quem será calculado o fatorial.
- Uma chamada de função que retorne um valor deve ser feita sempre dentro de um comando onde o valor de retorno será imediatamente utilizado.
- Exemplos da chamada da função dentro do programa seriam algo como:

```
a=fat(4);
a=fat(b);
printf("O fatorial de %d = %d\n",b,fat(b));
```

- Funções podem chamar outras funções. Nesse caso, a função chamada deve estar declarada antes do ponto onde ela é chamada.
- Caso isso não seja possível, pode-se usar um **protótipo** que é simplesmente a linha do cabeçalho, sem o corpo da função, que deve estar desenvolvido mais adiante.

- Exemplo:

```
int poi()
        {
        printf("Oi");
        ptchau();
        }
int ptchau()
        {
        printf("Tchau");
        poi();
        }
```

- Como a função poi chama a ptchau, e a ptchau chama a poi, não é possível ordená-las de modo que a chamada da função fique após a declaração da mesma. A solução para isso é o uso de um protótipo de uma das funções:

- Exemplo:

```
int ptchau(); // protótipo da ptchau
int poi()
        {
        printf("Oi");
        ptchau();
        }
int ptchau()
        {
        printf("Tchau");
        poi();
        }
```

- Uma função pode utilizar variáveis declaradas dentro dela, chamadas **variáveis locais**, mas pode também utilizar variáveis declaradas fora de qualquer função. Essas variáveis são chamadas de **variáveis globais** e podem ser utilizadas para passar informações entre funções.
- Variáveis locais só podem ser referenciadas dentro da função onde são declaradas.
- Ex:

```
#include <stdio.h>
int i,b; // variáveis globais
int fat(int n)
{
int f=1; // f é local a fat
for (i=1;i<=n;i++) f=f*i;
// como i não foi declarado em fat,
// é utilizado o i global
return f;
}
int main()
{
int x; // x é local a main
scanf("%d",&a); // a é global
printf("O fatorial de %d = %d\n",a,fat(a));
}
```

- O comando **return** é utilizado para retornar a execução para o ponto onde a função foi chamada.
- Ele pode ser executado em qualquer ponto da função e causa o encerramento imediato da execução da função.
- Se a função deve retornar algum valor, o valor a ser retornado deve ser especificado no comando return.
- Em algumas situações uma função não deve retornar nenhum valor.
- Nesse caso ela deve ser especificada como função do tipo **void**.

- Por exemplo, a uma função que receba um valor N e escreva os números de 1 a N:

```
void esc1aN (int N)
{
        int i;
        for (i=1;i<=N;i++)
                printf("%d ",i);
        return;
}
```

- E a chamada de uma função sem valor de retorno é feita como se fosse um comando da linguagem. Por exemplo, a função anterior seria chamada

```
esc1aN(5);

for (j=1;j<5;j++) esc1aN(j);
```

Exercícios:

- 1) Faça uma função que receba o dia, mês e ano de nascimento de uma pessoa, e o dia, mês e ano atual, e retorne sua idade.
- 2) Faça uma função que receba um valor N e retorne o primeiro divisor maior que 1 do valor N.
- 3) Faça uma função que receba um valor N e retorne a quantidade de divisores do valor N.
- 4) Usando uma das duas funções anteriores, faça uma função que recebe um valor N e verifique se ele é primo, retornando 1 se ele é primo, e 0 se não for primo. Use essa função para escrever um programa que escreva os 100 primeiros números primos.

- 5) Usando a função anterior, faça uma função proxprimo(int N) que receba um valor N e retorne primeiro valor primo maior que N. Use essa função para escrever os primeiros 100 números primos.
- 6) Faça um programa que escreva os 50 primeiros números primos.
- 7) Faça um programa que leia 2 números N1 e N2 e escreva a soma dos números primos entre N1 e N2 (incluindo N1 e N2 se algum deles for primo)..
- 8) Faça um programa que leia 2 números N1 e N2 e escreva o produto dos números primos entre N1 e N2 (incluindo N1 e N2 se algum deles for primo).

- 9) Faça um programa que leia um número N e escreva os N primeiros números primos maiores que 100.
- 10) Faça um programa que leia um número inteiro N e escreva o maior número primo menor do que N.
- 11) Faça um programa que leia um número inteiro N e escreva o menor número primo maior do que N.
- 12) Um número primo é um número natural maior que 1, que é divisível somente por 1 e por ele mesmo. Faça um programa que leia um número inteiro N e escreva o número primo mais próximo a ele. Se N for primo, considere que o mais próximo é o próprio N. Se houver dois números à mesma distância, escreva os dois em ordem crescente.

- 13) A conjectura de Goldbach diz que todo número par maior que 2 pode ser representado como a soma de dois números primos. Assim, 4=2+2, 6=3+3, 8=3+5... Faça um programa que leia um número N, par, e escreva, em ordem crescente, os dois números primos que o compõem. No caso de haver mais de um par de números (p.ex:20=3+17 e 20=7+13) escreva o par que tiver o menor número primo.
- 14) Faça uma funçao que receba o valor de dois resistores e um terceiro parâmetro definindo o tipo de associação dos resistores (0 para em série, 1 para em paralelo) e retorne o valor da associação dos dois resistores.
- 15) Faça uma função que receba um vetor v[10] e retorne o valor do maior elemento do vetor.
- 16) Faça uma função que receba um vetor v[10] e escreva todos os elementos do vetor.

- 17) Faça uma função que receba um vetor v[10] e ordene-o em ordem crescente.
- 18) Faça uma função que receba dois vetores v[10] e w[10] e verifique se os vetores são iguais, retornando 1 se são iguais, 0 se tem pelo menos um elemento diferente
- 19) Faça uma função que receba uma matriz M[10][10] e retorne o valor do maior elemento da matriz.
- 20) Faça uma função que receba uma matriz M[10][10] e retorne a soma dos elementos da diagonal principal.
- 21) Faça uma função que receba uma matriz M[10][10] e retorne a soma dos elementos da diagonal secundária.

- 22) Faça um programa que escreva os primeiros 100 dígitos cuja soma dos dígitos formantes é 10.
- 23) O número de combinações de N diferentes objetos em grupos de P objetos é dado por $\frac{N!}{P!(N-P)!}$. Faça uma função que receba uma quantidade N de objetos e o tamanho P dos grupos a serem formados, e calcule e retorne a quantidade de grupos que podem ser formados.
- 24) O MDC (máximo divisor comum) entre dois números n1 e n2 é o maior número que é divisor de n1 e de n2. Faça uma função que receba dois números e escreva seu MDC.

Ponteiros

- Um ponteiro é uma variável que contém um endereço de memória. Esse endereço é normalmente uma posição de outra variável na memória ou o endereço de uma estrutura em memória alocada dinâmicamente (durante a execução).
- Usam-se ponteiros principalmente para:
  - Fazer passagem de parâmetros por referência em funções;
  - Acessar estruturas alocadas dinamicamente;

- Uma declaração de ponteiros consiste no formato:

  $<$ *tipo* $>$ *<nome da variável>

- onde tipo é qualquer tipo válido em C.
- Exs:

```
int *pta; // pointer para valor inteiro
float *ptf; // pointer para float
```

- O tipo base do ponteiro define que tipo de variáveis o ponteiro pode apontar.
- Basicamente, qualquer tipo ponteiro pode apontar para qualquer lugar, na memória.
- Mas, para a aritmética de ponteiros é feita através do tipo base.
- Os operadores utilizados para trabalhar com ponteiros são * e &.
- O *, quando colocado antes do ponteiro em uma referência, referencia a posição de memória apontada por ele.
- Ex:
- int *a;
- *a=3; // a posição de memória apontada por a recebe 3

- E o operador &, aplicado a uma variável, retorna o endereço dela.
- Ex:

```
int a, *pta;
pta=&a;      // pta recebe o endereço da variável a.
*pta=5; //    a variável apontada por pta (ou seja, a
```

```
1) Seja o seguinte trecho de programa:
int i=3,j=5;
int *p, *q;
p = &i;
q = &j;
```

- Qual é o valor das seguintes expressões ?
- a) p = = &i;
- b) *p - *q
- c) **&p
- d) 3* - *p/(*q)+7
- resposta: 1, -2, 3, 6

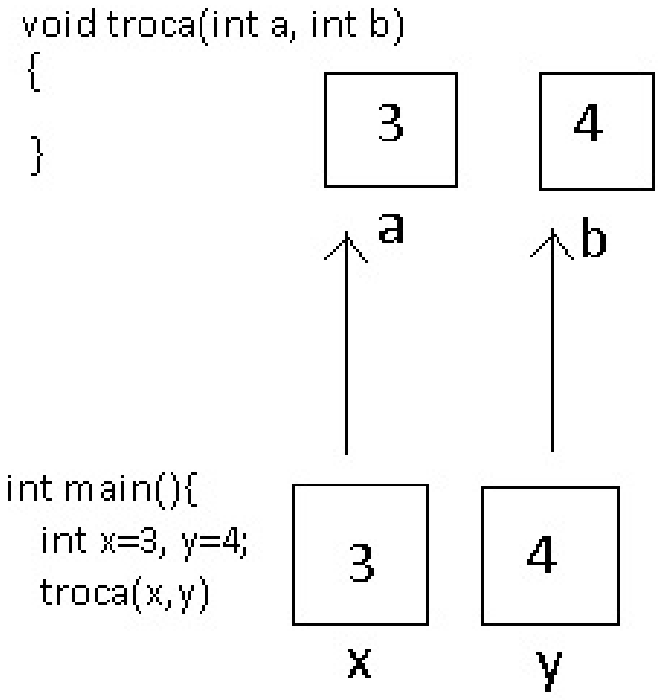
Tipos de passagem de parâmetros

- Existem diversas formas de passagem de parâmetros em funções, e cada linguagem define que formas de passagem a linguagem oferece.
- O que será escrito (valores de **a** e **b**) no código a seguir?

```
void troca(int x, int y)
{
int aux;
aux=x;
x=y;
y=aux;
}
int main()
{
int a,b;
a=3;
b=5;
troca(a,b);
printf("%d %d\"n,a,b);
}
```

Passagem por valor

- No código anterior, apesar de **x** e **y** terem sido trocados entre si dentro da função, os valores de **a** e **b** ao retornar do procedimento permaneceram inalterados.
- Isso ocorre porque, na linguagem C, quando é feita uma chamada de função, os parâmetros formais (declarados no cabeçalho) não ocupam a mesma posição de memória dos parâmetros atuais (os parâmetros enviados) e na chamada é passada uma cópia do valor do parâmetro, então as operações feitas sobre o parâmetro dentro da função são feitas sobre os parâmetros formais, e não tem efeito nenhum na variável passada.
- Essa forma de passagem de parâmetro, que é a forma utilizada em C, se chama **passagem por cópia** ou **passagem por valor**, e é utilizada para a passagem de **parâmetros de entrada**, ou seja, parâmetros pelos quais não retornam resultados.

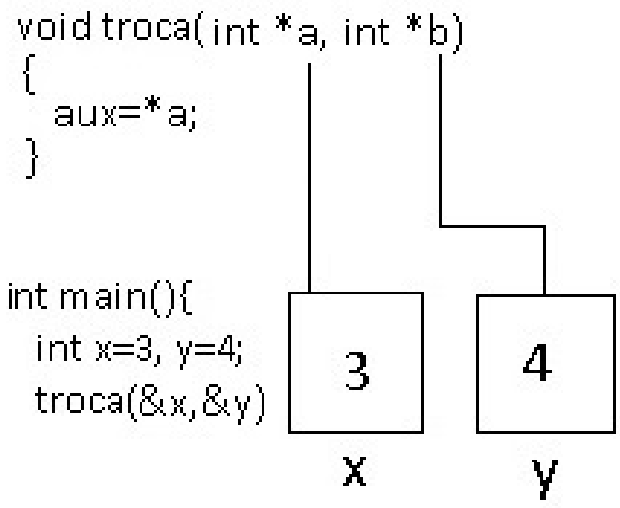


- Ao fazer a chamada são criadas a e b e após a cópia dos valores dos parâmetros atuais para os formais, não há mais ligação entre eles.
- Alterações nos parâmetros formais não tem efeito nenhum nos parâmetros atuais.

Passagem por referência (ou por endereço)

- Quando se deseja que as variáveis passadas como parâmetros tenham seus valores alterados dentro da função ou procedimento, utiliza-se a **passagem por referência** (ou **passagem por endereço**).
- Nesse tipo de passagem, ao invés de ser passada uma cópia do valor do parâmetro, é passado o endereço de memória da variável, de modo que alterações no parâmetro dentro da função ocorrem, realmente, na variável enviada como parâmetro.

- Para efetuar uma passagem por referência, o parâmetro formal deve ser declarado como um pointer para o tipo de valor a ser recebido (utilizando o operador *), e ao chamar a função envia-se o endereço da variável a ser enviada (utilizando-se o operador &).
- Por isso que no scanf as variáveis devem ser passadas com & e no printf não se usa o &.
  - A passagem de parâmetros do scanf é por referência e a passagem dos parâmetros do printf é por valor.
- Ex: void troca(int *x, int *y)



- Ao fazer a chamada **a** e **b** recebem os endereços dos parâmetros enviados, e através desses endereços acessam diretamente as variáveis enviadas.
- Alterações nas variáveis apontadas pelos parâmetros formais são efetuadas diretamente sobre os parâmetros atuais.

- Dentro da função, toda a referência à variável apontada pelo parâmetro formal deve ser feita através do operador *.

```
void troca(int *x, int *y)
{
int aux;
aux=*x; // leia-se "aux recebe o valor apontado por x"
*x=*y; // a posição apontada por x recebe o valor apontado por y
*y=aux; // a posição apontada por y recebe o valor em aux
}
int main()
{
int a,b;
troca(&a,&b); // envia os endereços de a e b
}
```

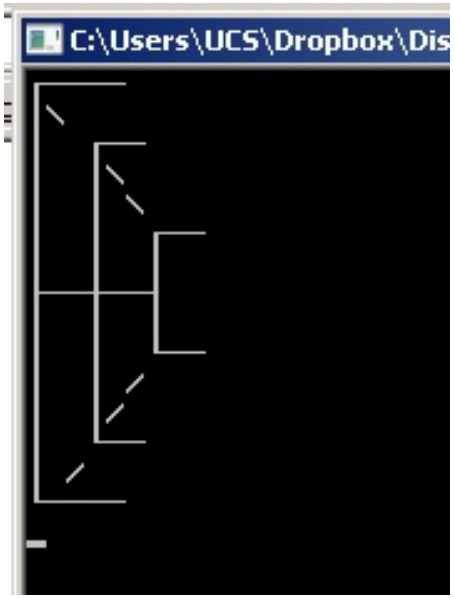
- A passagem por referência é utilizada para a passagem de parâmetros de saída, ou seja, parâmetros enviados para receber resultados de processamento.
- Normalmente são utilizados quando se precisa retornar mais de um valor, o que impede o uso de função, que só pode retornar um valor.
- A passagem por referência obriga que os parâmetros passados sejam variáveis, ao contrário da passagem por valor, em que os valores passados na chamada podem ser constantes ou expressões aritméticas

- Passagem de vetores em matrizes é sempre por referência, ou seja, sempre é passado o endereço inicial do vetor ou matriz.
- Para a passagem de vetores e matrizes, não deve ser usado o &.

- 1) Faça uma função que receba uma hora, minuto e segundo e retorne, nos mesmos parâmetros, a hora, minuto e segundo correspondentes ao segundo seguinte.
- 2) Faça uma função que receba uma data, formada por dia, mês e ano, e retorne, nos mesmos parâmetros, a data do dia seguinte. Considere que ano bisexto é aquele divisível por 4. Considere também que abril, junho, setembro e novembro tem 30 dias, fevereiro tem 28 (29 se o ano for bisexto) e todos os outros meses tem 31 dias.
- 3) Faça uma função que receba um vetor M[10] e retorne, no mesmo parâmetro, o vetor ordenado em ordem crescente.

- 4) Faça uma função que receba dois vetores V[10] e W[10] e retorne um terceiro vetor Z com os elementos que estão em V e W (interseção). Retorne também, em um quarto parâmetro o número de elementos que foram colocados em Z.
- 5) Faça uma função que receba dois vetores V[10] e W[10] e ordene os dois vetores, gere a seguir um terceiro vetor Z[20] com todos os elementos de V e W, também ordenado, retornando o vetor Z.

Desenho de caracteres gráficos em modo texto




```
printf("%c%c%c%c%c\n",218,196,196,196,196);
printf("%c%c \n",179,92);
printf("%c %c%c%c \n",179,218,196,196);
printf("%c %c%c \n",179,179,92);
printf("%c %c %c \n",179,179,92);
printf("%c %c %c%c%c \n",179,179,218,196,196);
printf("%c %c %c \n",179,179,179);

printf("%c%c%c%c%c%c%c\n",195,196,196,197,196,196,180);
printf("%c %c %c \n",179,179,179);
printf("%c %c %c%c%c \n",179,179,192,196,196);
printf("%c %c %c \n",179,179,47);
printf("%c %c%c \n",179,179,47);
printf("%c %c%c%c \n",179,192,196,196);
printf("%c %c \n",179,47);
```

Atividades no site hackerrank.com

- Acesse o site: https://www.hackerrank.com/
- Efetue um cadastro no site ("sign up", canto superior direito da tela)
- Resolva os primeiros exercícios de "warm up" (aquecimento) submetendo-os até obter "accepted" em cada um, categoria "easy".
- Resolva o exercício Extra Long Factorials, primeiro exercício da categoria "medium" (para isso precisaremos de algumas funções extras...).

Exercícios com números realmente grandes

- Faça uma função que receba duas strings, cada uma com um número até 20 dígitos, e retorne (como parâmetro) uma string com o resultado da soma de uma string pela outra.
  - Se a primeira string for "123" e a segunda for "12300", a string retornada deve ser "12423".
- Faça uma função que receba duas strings, cada uma com um número até 20 dígitos, e retorne (como parâmetro) uma string com o resultado da multiplicação de uma string pela outra.
  - Que funções auxiliares poderiam ser implementadas para simplificar a função acima?

Funções recursivas:

- Uma função é dita recursiva quando contem, em seu corpo, uma chamada a si mesma. São utilizadas quando é possível decompor o problema a ser resolvido em problemas maiores, um dos quais é semelhante ao problema inicial.
- Ex: fatorial(n)
- n!=n*(n-1)*(n-2)*....*1
- Uma forma recursiva de definir o fatorial é

$$fat(n) = \begin{cases} 1 & p/n <= 1 \\ n * fat(n-1) & p/n > 1 \end{cases}$$

- E uma implementação em C pode ser:

```
int fat(int n)
{
    if (n<=1) return 1;
    return n*fat(n);
}
```

A soma dos números de 1 a n pode ser definida de forma recursiva como

$$Soma(n) = \begin{cases} n + soma(n-1) & p/n > 1 \\ 1 & p/n = 1 \end{cases}$$

A soma dos números ímpares de 1 a n pode ser definida de forma recursiva

$$Somaimp(n) = \begin{cases} n + somaimp(n-1) & p/n \quad impar \quad e \quad n > 1 \\ somaimp(n-1) & p/n \quad par \quad e \quad n > 1 \\ 1 & p/n = 1 \end{cases}$$

Quicksort

Código do particionamento do quicksort

```
int separa( int v[], int p, int r)
    {
    int c = v[p], i = p+1, j = r, t;
    while (1) {
        while (i <= r && v[i] <= c) ++i;
        while (c < v[j]) - -j;
        if (i >= j) break;
        t = v[i], v[i] = v[j], v[j] = t;
        ++i; - -j;
        }
    v[p] = v[j], v[j] = c;
    return j;
    }
```

Exercícios de recursão sem vetores

1) Faça uma função recursiva que receba dois números a e b , $a \leq b$, e retorne a soma dos números ímpares entre a e b.

2) Faça uma função recursiva que receba um valor N e escreva todos os valores de 1 a N em ordem crescente.

3) Faça uma função recursiva que receba um valor N e escreva todos os valores de 1 a N em ordem decrescente.

4) Faça uma função recursiva que receba um valor N e escreva todos os seus divisores.

5) Faça uma função recursiva que receba um valor N e retorne a soma de seus divisores.

6) Faça uma função que receba um número N e retorne:

0 - Se não é primo

1 - Se é primo

Use as funções auxiliares que quiser, mas não utilize iteração em nenhuma delas.

7) Faça uma função recursiva que receba dois valores A e B e escreva os números primos entre A e B (incluindo A e B).

8) Faça uma função recursiva que receba dois valores A e B e escreva os números primos entre A e B (incluindo A e B).

9) Faça uma função recursiva que receba dois valores A e B e retorne a soma dos primos entre A e B (incluindo A e B).

10) Faça uma função recursiva que receba um valor N e escreva seus dígitos invertidos.

11) Faça uma função recursiva que receba 3 números a, b e c, sendo $a < b$, e retorne a soma dos divisores de c entre a e b.

12) Faça uma função recursiva que leia 50 números e retorne o maior deles.

13) Faça uma função recursiva que leia 50 números e retorne:

0 - Se todos são positivos
1 - Se pelo menos um deles é negativo

14) A série de Fibonacci é uma seqüência de números onde os primeiros são 0 e 1 e a partir destes, qualquer termo da série é dado pela soma dos dois anteriores. Faça uma função recursiva que receba a ordem de um termo e retorne o valor do termo.

15) Faça uma função recursiva que leia 50 números e retorne:

0 - Se todos são ímpares.
1 - Se pelo menos um deles é par.

Obs : A função deve ler 50 números, nem mais nem menos.

Exercícios de Recursão com vetores

1) Faça uma função recursiva que receba um vetor de 100 posições e retorne o somatório dos elementos pares do vetor.

2) Faça uma função recursiva que receba um vetor de 100 posições e retorne a posição do menor elemento do vetor.

3) Faça uma função recursiva que receba um vetor de 100 posições e retorne o valor do menor elemento do vetor.

4) Faça uma função recursiva que receba um vetor de 100 posições e retorne o valor do maior e o valor do menor elemento do vetor (como parâmetros de retorno).

5) Faça uma função recursiva que receba um vetor V(100) e retorne a média dos valores do vetor.

6) Faça uma função recursiva que leia uma matriz M(10,10) e retorne a matriz lida.

7) Faça uma função recursiva que receba um vetor V(100) e troque o 1 elemento com o 51, o 2 com o 52 até o 50 com o 100.

8) Faça uma função recursiva que receba um vetor A(100) e inverta o vetor.

9) Faça uma função recursiva que receba um vetor A(100) e retorne o número de elementos nulos do vetor.

10) Faça uma função recursiva que receba um vetor de 100 números inteiros e retorne a soma dos elementos do vetor. Mostre como seria a chamada desta função em um programa.

11) Faça uma função recursiva MAIOR que receba uma matriz numérica 10x10 e retorne o maior elemento da matriz. Utilize esta função em um programa que leia uma matriz A(10,10) e escreva seu maior elemento.

12)Faça uma função recursiva que receba um vetor de 100 números inteiros e retorne a soma dos elementos do vetor. Mostre como seria a chamada desta função em um programa.

13) Faça uma função recursiva que, sem utilizar vetores, leia 50 valores e os escreva na ordem contrária à que foram lidos.

Exercícios de recursão com Matrizes

1)Faça uma função recursiva que leia uma matriz M(10,10) e retorne a matriz lida.

2)Faça uma função recursiva que receba um vetor A(100) e inverta o vetor.

3)Faça uma função recursiva que receba um vetor A(100) e retorne o número de elementos nulos do vetor.

4)Faça uma função recursiva que receba uma matriz 10x10 com 0's e 1's, duas posições da matriz (linha e coluna de cada) e retorne:

1 - Se há um caminho de 0's entre as duas posições
0 - Se não há um caminho de 0's entre as duas posições

5)Faça uma função recursiva que receba duas matrizes de inteiros M(50,50) e N(50,50) e retorne:

0 - Se as matrizes são iguais

1 - Se as matrizes são diferentes

6)Faça uma função recursiva MAIOR que receba uma matriz numérica 10x10 e retorne o maior elemento da matriz. Utilize esta função em um programa que leia uma matriz A[10][10] e escreva seu maior elemento.

Desafios

- 32) Faça uma função

  void Remove(int M[20][20],int Saida[20][20],int Ordem,int Linha,int Coluna)

- que receba de entrada uma matriz M[20][20], a ordem de uma matriz quadrada armazenadaa nas linhas e colunas iniciais de M, e os números de uma linha LIN e uma coluna COL, e remova de M a linha LIN e a coluna COL, retornando a matriz resultante na matriz Saida.

33) O determinante de uma matriz é igual à soma dos produtos de cada elemento a(i,j) de uma linha i pelo determinante da matriz resultante da remoção da linha i e coluna j por $(-1)^{(i+j)}$. E o determinante de uma matriz 1 x 1 é o próprio elemento. Faça uma função recursiva que receba uma matriz M[20][20] e a ordem da matriz quadrada que ela contem, e retorne o seu determinante.

34) O menor complementar de um elemento a(i,j) em uma matriz A[N][N] é o determinante da matriz de ordem [N-1]x[N-1] obtida da eliminação da linha i e da coluna j. Faça uma função que receba uma matriz A[20][20], a ordem da matriz quadrada que ela contem nas primeiras linhas e colunas, e índices i,j, e retorne o menor complementar do elemento A[i][j].

35) O Complemento algébrico de um elemento a(i,j) em uma matriz A[N][N] é o produto de $(-1)^{(i+j)}$ pelo seu menor complementar. Faça uma função que receba uma matriz A[20][20], a ordem da matriz quadrada que ela contem nas primeiras linhas e colunas, e índices i,j, e retorne o complemento algébrico do elemento A[i][j].

36) A matriz transposta de uma matriz A é a matriz obtida trocando-se entre si cada elemento A(i,j) com o elemento A(j,i). Faça uma função que receba uma matriz A[20][20], a ordem da matriz quadrada que ela contem nas primeiras linhas e colunas, e retorne uma matriz T[20][20] com a matriz transposta de M.

37) Chama-se matriz adjunta da matriz A à transposta da matriz que se obtem de A por substituição de cada um dos seus elementos pelo respectivo complemento algébrico. Faça uma função que receba uma matriz A[20][20], a ordem da matriz quadrada que ela contem nas primeiras linhas e colunas, e retorne uma matriz ADJ[20][20] com a matriz adjunta de A.

38) A matriz inversa de uma matriz quadrada A pode ser obtida pelo produto do inverso do determinante da matriz A pela matriz adjunta de A. Faça uma função que receba uma matriz A[20][20], a ordem da matriz quadrada que ela contem nas primeiras linhas e colunas, e retorne uma matriz INV[20][20] com a matriz inversa de A. Efetue o produto matricial de A pela inversa obtida para verificar se o cálculo da inversa está correto.

39) (Sistema de Cramer) Um sistema de equações pode ser representado na forma A x = b, onde A é a matriz com os coeficientes das equações, x é o vetor das incógnitas e b é o vetor de termos independentes. Cada elemento i do vetor x pode ser calculado por det(Ai)/det(A), onde Ai é a matriz obtida pela substituição da coluna i pelo vetor de termos independentes. Faça uma função que receba uma matriz M[3][3], um vetor b[3] e retorne o vetor x, com a solução do sistema.

40)Faça uma função que receba um vetor e retorne o vetor ordenado pelo Bubblesort. Converta os dois laços de repetição do bubblesort em um único laço de repetição. Transforme o laço de repetição em uma chamada recursiva.

- Medição de tempo em trechos de código pode ser feita utilizando a função time( ) da biblioteca time.h, que retorna o número de segundos transcorridos desde 1/1/1970.
- Pode ser utilizada da seguinte forma:

```c
#include <time.h>
#include <stdlib.h>
#include <stdio.h>

int main(void)
{
time_t t0,t1;
t0=time(NULL);

// trecho de código a ser medido

t1 = time(NULL);
printf("tempo:%d\n",t1-t0);
}
```

- Faça um programa que execute um bubblesort para tamanhos de vetores de 1000 a 10000 em intervalos de 1000, escrevendo para cada ordenação a quantidade de elementos e o tempo levado. Antes de cada ordenação inicialize o vetor com valores randômicos.
- Para gerar valores randômicos utiliza-se a função rand( ), da biblioteca stdlib.h. Cada chamada da função rand( ) retorna um valor inteiro entre 0 e RAND_MAX (constante definida em stdlib). Para inicializar o gerador de números randômicos no início do programa chama-se a função
- srand (time(NULL));
- Ex:

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

int main ()
{
srand ( time(NULL) );

printf ("Um␣numero␣entre␣0␣e␣RAND_MAX␣(%d):␣%d\n", RA
printf ("Um␣numero␣entre␣0␣e␣99:␣%d\n", rand()%100);
printf ("Um␣numero␣entre␣20␣e␣29:␣%d\n", rand()%10+20

return 0;
}
```

1) Faça um programa que leia dois vetores de 20 elementos cada (ou preencha com valores randômicos ou inicialize-os na declaração), e ordene em ordem crescente cada um dos dois vetores separadamente. Gere a seguir um vetor contendo todos os elementos dos dois vetores, também em ordem crescente e escreva o vetor ordenado. Faça o programa mais eficiente que puder.

Mergesort... Determinante por Cramer... Torre de Hanoi... Caça-palavras...

Mensagens de erro do Dev C++

- **'a' undeclared (first use this function)**
  - A variável 'a' não foi declarada
- **expected ';' before "b"**
  - Faltou um ponto-e-vírgula no comando anterior
- **Warning - Converting to 'int' from 'double'**
  - Está sendo atribuído um valor real a uma variável inteira
  - Um 'Warning' é uma mensagem de advertência, mas que não impede o programa de ser executado. Normalmente indicam situações de erro em potencial.

Provas Anteriores