Universidade Federal de Juiz de Fora
Faculdade de Engenharia e Arquitetura
Programa de Educação Tutorial

Apostila de Introdução à Linguagem C

Por Marcela Rocha Tortureli de Sá
Bolsista do grupo PETCivil

Agradecimentos:

Agradeço em especial à Professora Ana Paula Couto com quem tive aula e após ter finalizado a cadeira me ajudou na formulação desta apostila.

Índice:

1-Introdução:

O que é C?

C é uma **linguagem de programação de computadores**: É possível usá-la para criar um conjunto de instruções para que o computador possa executar. Isso significa que você pode usá-la para criar listas de instruções para um computador seguir. A linguagem C é uma das milhares de linguagens de programação atualmente em uso. Desenvolvida em 1972 por Dennis Ritchie no Bell Lab para uso no sistema operacional Unix, foi amplamente aceita por oferecer aos programadores o máximo em controle e eficiência. Existe há várias décadas e ganhou ampla aceitação por oferecer aos programadores o máximo em controle e eficiência.

A linguagem C foi criada com o objetivo principal em mente: facilitar a criação de programas extensos com menos erros, recorrendo ao paradigma da programação algorítmica ou procedimental. [, mas sobrecarregando menos o autor do compilador, cujo trabalho complica-se ao ter de realizar as características complexas da linguagem.]

Motivação:

Por ser uma linguagem poderosa, robusta, flexível e madura, conhecer todos os seus detalhes, "truques" e "armadilhas" requer um estudo criterioso e profundo. Mas o C é uma linguagem de sintaxe simples e elegante que permite rápido entendimento pelo programador iniciante. Desde sua criação, o C tornou-se uma linguagem popular tanto entre programadores profissionais quanto entre os iniciantes. É a linguagem de programação preferida para o desenvolvimento de sistemas e softwares de base, apesar de também ser usada para desenvolver programas de computador. No meio acadêmico, é amplamente utiliza para desenvolvimento de pesquisas científicas e como instrumento de aprendizado para o desenvolvimento de algoritmos.

A linguagem C é amplamente utilizada no meio acadêmico.

Somente para exemplificar o quanto a linguagem C é difundida, podemos citar alguns *softwares* escritos em C: Matlab, Windows, Kernel do Linux, Nedit (editor de texto), Octave (similar do Matlab em código aberto), o servidor de vídeo RI, Skype (escrito em C/C++) e Network Simulator (NS2).

2-Sintaxe:

A sintaxe são regras detalhadas para cada construção válida.
Para que o programa criado seja executado de maneira correta, é necessário seguir a sintaxe própria da linguagem.
Identificadores: É o nome dado pelo programador a um objeto, que será utilizado para que este possa ser manipulado durante o programa.

Exemplo: a, tamanho, idade, SomaVetores, ..

Ao declarar um identificador você dá um tipo que determina como os valores de dados serão representados, que valores podem expressar, e que tipo de operações você pode executar com estes valores. È importante ressaltar que em C letras maiúsculas diferem das minúsculas (a linguagem é Case Sensitive).

Tipos definem as propriedades dos dados manipulados em um programa. Variáveis ficam armazenadas de acordo com suas propriedades de informação.

Int – para armazenamento de qualquer número inteiro negativo, nulo ou positivo. Ocupa 2 bytes (long int 4 bytes) de memória. Exemplo:- 5, 0, 2

Float – para armazenar qualquer número real negativo, positivo ou nulo. Ocupa 4 bytes (doublé 8 bytes) de memória. Exemplo:- 1.78, 98, 0

Char – usada para se armazenar quaisquer letras e números (conjunto de caracteres alfanuméricos). Ocupa 1 byte de memória. Exemplo: ”A”,”B”

As **declarações** expressam as partes do programa, podendo dar significado a um-identificador, alocar memória, definir conteúdo inicial, definir funções.

As **funções** especificam as ações que um programa executa quando roda.

Uma função importante em todo programa em C é a função *main* (cuja tradução é *principal*). Esta será sempre a primeira função do programa a ser executada.

```
main ( )

{

}
```

2-Variáveis:

É um local na memória principal, isto é, um endereço que armazena um conteúdo (informação) que pode ser modificado.

Em C, não é possível ter variáveis que comecem com dígito e espaços não são permitidos.

Exemplos de nomes de variáveis indevidas: 2w, peso do aluno, sal/hora.

Observação:Em C usualmente são utilizadas variáveis em minúsculo e constantes em maiúsculos.

3-Operadores:

=	atribuição
!=	diferente
<=	Menor igual
>=	Maior igual
==	igual
%	Resto da divisão entre inteiros

Para as variáveis, devem ser feitas no início do programa (ou de um bloco) as declarações (de tipo) de variáveis.

Declarada uma variável, toda vez que ela for referenciada em qualquer comando do programa, o computador vai trabalhar com o conteúdo de seu endereço, que é o valor da variável.

Uma variável não pode ter o mesmo nome de uma palavra-chave de C, como por exemplo: main, cout, int, float, char, short, return, case, void.

As variáveis só podem armazenar informações ou dados sempre de um mesmo tipo (inteiro, real, caractere ou char).

Forma de declaração de variáveis em linguagem C:

<tipo> <nome_var>;

ou

<tipo> <nome_var1>, <nome_var2>, ... ,<nome_varn>;

Exemplo: int numero,soma;

É importante não esquecer que quando se declara uma variável, um espaço num determinado endereço na memória é alocado para armazenar um dado que obrigatoriamente tem que ser do mesmo tipo que o da variável.

Toda e qualquer variável deve ser declarada e sua declaração deve ser feita antes de sua utilização no programa.

Operador de atribuição:
<variavel> = <expressão>;
Exemplos: a = 5;

Operadores Aritméticos:

Aritméticos	Tipo	Operação	Prioridade
+	Binário	Adição	5
-	Binário	Subtração	5
%	Binário	Resto da divisão	4
*	Binário	Multiplicação	3
/	Binário	Divisão	3
++	Unário	Incremento	2
--	Unário	Decremento	2
+	Unário	Manutenção do sinal	1
-	Unário	Inversão do sinal	1

Sendo que varia de 1 para a prioridade máxima a 5.

Exemplo:

a=a+b;
a = 4 * 2 + 3;

Observação:
a++ é similar a **a = a + 1;**
b-- é similar a **b = b -1;**

Outros Operadores:

Relacionais		Lógicos	
>	Maior que	&&	And (e)
>=	Maior ou igual	||	Or (ou)
<	Menor que	!	Not (não)
<=	Menor ou igual		
==	Igual		
!=	Diferente		

4-Comandos de Entrada e Saída
4.1-Comando de Imprensão printf

Através da função pré-definida *printf()*, cujo protótipo está contido também no arquivo *stdio.h*. Sua sintaxe é a seguinte:

printf("Expressão" , lista de argumentos);

Sintaxe:
<expressão> Mensagens que serão exibidas.
<lista de argumentos> pode conter identificadores de variáveis, expressões aritméticas ou lógicas e valores constantes**.**

Impressão de Tipos de Dados

Código	Tipo	Elemento armazenado
%c	char	um único caractere
%d ou %i	int	um inteiro
%f	float	um número em ponto flutuante
%lf	double	ponto flutuante com dupla precisão
%e	float ou double	um número na notação científica
%s	.....	uma cadeia de caracteres

Primeiro Programa em C!

```
#include <stdio.h>

int main()
{
    printf("%s está a %d milhões de milhas do sol","Vênus",67);
}
```

```
#include <stdio.h>

int main( )
{
  printf(“Valor inteiro atribuído foi %d para o caracter  %c e um float foi de %f “,99,a,1.45);
}
```

```
 #include <stdio.h>
int  main()
 {
      printf(“Se quisesse imprimir uma string : %s“,”Minha string!”);
  }
```

Impressão de Códigos Especiais

Código	Ação
\n	Leva o cursor para a próxima linha
\t	Executa uma tabulação
\b	Executa um retrocesso
\f	Leva o cursor para a próxima página
\a	Emite um sinal sonoro (*beep*)
\"	Exibe o caractere "
\\	Exibe o caractere \
%%	Exibe o caractere *%*

```
 int main( )
{
Printf(“\n”);
}
```

Por **default**, a maioria dos compiladores C exibem os números de *ponto flutuante* com seis casas decimais.

Para alterar este número podemos acrescentar .n ao código de formatação da saída, sendo *n* o número de casas decimais pretendido.

```
#include <stdio.h>

int main()
{
 printf("Default: %f \n",3.1415169265);
 printf("Uma casa: %.1f \n",3.1415169265);
 printf("Duas casas: %.2f \n",3.1415169265);
 printf("Três casas: %.3f \n",3.1415169265);
 printf("Notação Cientifica: %e \n",3.1415169265);
}
```

Embora o número digitado possua 10 casas decimais imprimirá respectivamente, 6, 10, 1, 2,3e 10 casas decimais.

Exercícios

1-Fazer um programa que imprima o nome da sua cidade natal.
2-Faça um programa que imprima na primeira linha seu nome, na segunda sua idade e na terceira sua altura.
3-Faça um programa que imprima o numero 75.7632489 com 1,2 e 5 casas decimais.

4.2-Leitura de dados - scanf()

Ela é o complemento de printf() e nos permite ler dados formatados da entrada padrão (teclado). Sua sintaxe:

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

Exemplo:

```
int m;
scanf (“%d”,&m);
```

- %d indicativo do tipo, neste caso do tipo inteiro.
- &m operador utilizado para obter o endereço de memória da variável.

Exemplo:

```
int main ()
{
int idade;

Printf(“Digite a sua idade”);
Scanf(“%d”,&idade);
Printf(“A sua idade é %d”,idade);
Return 0;

}
```

Exercícios:

1-Faça um programa que peça ao usuário para digitar sua altura, leia o dado digitado e imprima o valor.
2-Faça um programa o qual o usuário digite seu peso e o de mais duas pessoas, imprima os valores digitados.

5-Comando Condicional – if/else

5.1-Comando condicional if

O comando if é uma estrutura de decisão que permite ou não que uma seqüência de comandos seja executada (ou não executada)., dependendo do resultado de uma condição pré-estabelecida. Sua sintaxe é:

```
if (<expressão>)
{
<sequência de comandos>
}
Ou
if (<expressão>) <único comando>;
```

A expressão sempre será avaliada logicamente (verdadeiro ou falso), correspondendo ao FALSO o valor zero (==0) e os demais ao VERDADEIRO (!=0).

Exemplo:
Definir qual é o menor número digitado pelo usuário:

```
 int main()
{
 int a,b,menor;
printf("Digite dois números inteiros");
 scanf("%d %d",&a,&b);
menor=a;
  if  a>b
 menor=b;
 printf("O menor número digitado foi %d",menor);
 return 0;
}
```

5.2-Comando Condicional – if else

O comando if pode decidir entre duas sequências de comandos qual vai ser a executada, tendo a seguinte sintaxe:

```
if (<expressão>)
{ // caso a expressão verificada retorne verdadeiro
<sequência de comandos>
}
else
{ // caso a expressão verificada retorne falso
<sequência de comandos>
}
```

Exemplo: Verificar a paridade de um número.

```
#include <stdio.h>
int main()
{
int x;
printf("Digite um número inteiro: ");
scanf("%d", &x);
if (x % 2 == 0)
printf("%d e' par \n", x);
else
printf("%d e' impar \n", x);
```

```
if <condição 1>
  {
 <comandos if1>
  if <condição 2>
        {
        <comandos if2>
        }
        else
             {
               <comandos else2>
             }
  }
else
  {
     <comandos else1>
      if <condição 3>
      {
       <comandos if3>
       }
   }
}
```

Exemplo:

```
int main()
{
int IDADE;
printf("Informe sua idade: ");
scanf("%d",&IDADE);
if (IDADE < 20)
 {
  if (IDADE < 13)
   printf("Infantil.");
   else
   printf("Adolescente");
 }
else (IDADE < 50)
printf("Adulto");
else printf("Idoso");
return 0;
}
```

Observe que as chaves não foram utilizadas nesse exemplo no segundo e terceiro else umas vez que dentro de cada else há apenas uma condição quando há mais que uma torna-se necessário a utilização de chaves.

Exercícios:

1. Faça um programa para ler dois números reais, faça a divisão do primeiro número pelo segundo (se o segundo for diferente de zero).

2. Faça um programa para ler dois números reais e verificar se ambos são maiores que zero. Caso positivo, informar "Valores são válidos". Caso contrário, informar "Valores inválidos".

6-Comando Switch:

Utilizado quando uma determinada variável pode ser igual a diferentes valores que se deseja avaliar a variável é testada sucessivamente contra uma lista de variáveis inteiras ou de caracteres. Depois de encontrar uma coincidência, o comando ou o bloco de comandos é executado.

Se nenhuma coincidência for encontrada o comando default será executado. O default é opcional. A seqüência de comandos é executada até que o comando break seja encontrado. (tradução de switch-escolha,de case-caso,de break-senão,de default-fim escolha).

Sintaxe:

```
switch (variável)
{
case constante1: <comandos>
break;
case constante2: <comandos>
break;
default: <comandos>
}
```

Exemplos:

```
int main()
{
int EPOCA;
printf("Digite o trimestre do ano em que estamos: ");
scanf("%d",&EPOCA);
switch (EPOCA)
{
case 1: printf("verão");
break;
case 2: printf("outono");
break;
case 3: printf("inverno");
break;
case 4: printf("primavera");
break;
default: printf("período inválido");
}
return 0;
}
```

Exercícios:

1-Qual é a saída do programa.

```
int main()
{
int x = 10, y = 3;
if (x < 10)
printf(“Primeira Saida ”);
else
if (y < 4)
if (x > 10)
printf(“Segunda Saida ”);
else
 printf(“Terceira Saida ”);
return 0;
}
```

2-Os funcionários de uma empresa receberam um aumento de salário: técnicos, 50%; gerentes,30%;demais funcionários, 20%. Escrever um algoritmo que receba como entrada o salário atual e o cargo do funcionário, e imprima o novo salário após o aumento.

3-Calculadora:Fazer um algoritmo para ler dois números e um dos símbolos das operações: +, -, * e /. Imprimir o resultado da operação efetuada sobre os números lidos.

7-Comando while:

Uma maneira possível de executar um laço é utilizando o comando while. Ele permite que o código fique sendo executado numa mesma parte do programa de acordo com uma determinada condição.

- o comando pode ser vazio, simples ou bloco
- ele é executado desde que a condição seja verdadeira
- testa a condição antes de executar o laço

Sintaxe:

```
while(condição)
{
<comandos>;
}
```

Exemplo:

```
int main()
{
int i;
i=0;
while (i<=10)
{
printf("Número %d\n",i);
i++;
}
return 0;
}
```

Exercícios:
1-Passar para linguagem C:
a)
Inicio
inteiro: NUM, QUADRADO;
{imprime uma msg e lê o 1o. numero inteiro}
imprima(“Digite um número inteiro: ”);
leia(NUM);
enquanto NUM ≠ 0 faça
QUADRADO ← NUM**2;
imprima (“O quadrado de ”, NUM, “eh”, QUADRADO);
imprima(“Digite um número inteiro: ”);
leia(NUM);
fim enquanto;
fim.

b)
inicio
inteiro: SOMA, CONT, NUM; {CONT representa o contador}
real: MEDIA;
{SOMA o acumulador}
{imprime uma msg e lê o 1o. numero inteiro}
imprima(“Digite um número inteiro: ”);
leia(NUM);
CONT □ 0;
SOMA □ 0;
enquanto NUM ≠ 0 faça
CONT □ CONT + 1; {contador}
SOMA □ SOMA + NUM; {acumulador}
imprima(“Digite um número inteiro: ”);
leia(NUM);
fim enquanto;
MEDIA □ SOMA/CONT; {e se CONT for zero?}
imprima(“A media eh ”,MEDIA);
fim.

8-Comando For:

O comando for é de alguma maneira encontrado em todas linguagens procedurais de programação.
Em sua forma mais simples, a incialização é um comando de atribuição que o compilador usa para estabelecer a variável de controle do loop. A condição é uma expressão de relação que testa a variável de controle do loop contra algum valor para determinar quando o loop terminará. O incremento define a maneira como a variável de controle do loop será alterada cada vez que o computador repetir o loop.
Primeira vez que executa o for:
• Inicialização
• Condição
• VERDADEIRO: Bloco de comandos
• FALSO: sai do for
Demais vezes:
• Incremento
• Condição
• VERDADEIRO: Bloco de comandos
• FALSO: sai do for

Sintaxe:
for(inicialização;condição;incremento)
<comandos>;

Exemplo:

```
int main ()
{
int i,idade;

for (i=0;i<=5;i++)
{
printf (“ Digite uma idade”);
scanf (“%d”,&idade);
printf (“A idade digitada foi %d \n”);
}
return 0;
}
```

Exercício:
1-Fazer um programa que imprima 10 números de matricula fornecidos um a um pelo usuário.

9-Vetores (arrays):

Em diversas situações os tipos básicos de dados (int, float, char, ...) não são suficientes para representar a informação que se deseja armazenar. Exemplo, uma palavra: “AULA”.
Existe a possibilidade de construção de novos tipos de dados a partir da composição (ou abstração) de tipos de dados primitivos. Esses novos tipos têm um formato denominado ESTRUTURA DE DADOS, que define como os tipos primitivos estão organizados.

Exemplo:
Definir a média das notas de 50 alunos de uma turma.

```
#define N 50  //constante simbólica
int main()
{   //lê as notas de N alunos de uma turma
    //e calcula a media da turma

    int i;  //variável de controle
    float nota, media, soma = 0;

    for (i=1; i<=N; i++)
    {
        printf("Digite uma nota:");
        scanf("%f",&nota);
        soma = soma + nota;
    }

    media = soma/N;
    printf("Media = %0.2f",media);
    return 0;
}
```

Toda a ocorrência de N será substituída por 50

Neste ponto; qual o valor da 1ª ou 5ª ou 40ª nota?

Mas se além da média quisesse fazer um programa em C para imprimir as notas lidas juntamente com a média da turma .

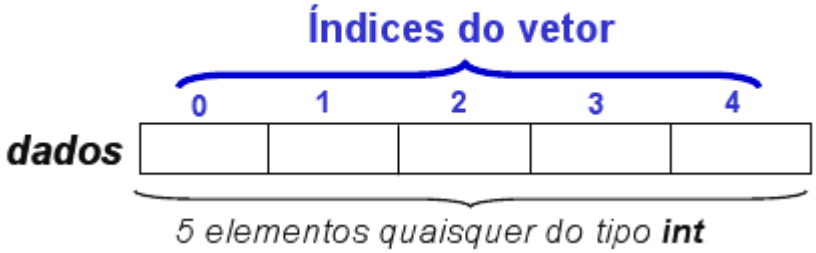
Quando uma determinada Estrutura de Dados for composta de variáveis com o mesmo tipo primitivo, temos um conjunto homogêneo de dados. Essas variáveis são chamadas de variáveis compostas homogêneas.

As variáveis compostas homogêneas unidimensionais são utilizadas para representar arranjos unidimensionais de elementos de um mesmo tipo, em outras palavras, são utilizadas para representar vetores.

sintaxe para declaração de uma variável deste tipo é:
tipo_primitivo identificador[qtde_elementos];

Exemplo:

int dados[5]; // vetor com 5 (0 a 4) elementos do tipo int



Sempre começará a contar a primeira posição do zero.
Para fornecer os elementos de cada posição do vetor é declarado da seguinte forma:

Exemplo:

float valores[4]={1.2,3.6,4.5,2.8};

valores

1.2	3.6	4.5	2.8
0	1	2	3

Se fosse
float dados[3]={1.2};
ou seja há menos valores do que o número de elementos do array, os elementos restantes são inicializados automaticamente com o valor zero.

Importante:
Os vetores não são inicializados automaticamente.
Declarar mais elementos do que defino causa erro de sintaxe.
Omitindo o tamanho do vetor ele terá o mesmo número de elementos digitados.
int n[ ] = {1, 2, 3, 4, 5};
ou seja 5 elementos,5 posições,índices do vetor variam de 0 a 4.

Cada um dos elementos de um vetor é referenciado individualmente por meio de um número inteiro entre colchetes após o nome do vetor.

Valores	2.6	8.9	1.3	4.5
	0	1	2	3

X = valores[1]; (8.9)
Y = valores[3]; (4.5)

Ambas as formas atribuem um valor ao elemento 0 (zero) do vetor valores:
i=0;
valores[0] = 6.7;
ou
valores[i]=6.7;

O programa a seguir inicializa os dez elementos de um array s com os valores: 2, 4, 6, ..., 20 e o imprime.

```
#include <stdio.h>
#define TAMANHO 10
int main()
{
int s[TAMANHO], j;
for (j=0; j<= TAMANHO - 1; j++)
{
s[j] = 2 + 2*j;
printf("Elemento  %d tem valor de %d \n”,j,s[j]);
}
return 0;
}
```

Exercícios:

1-Informar quais são os elementos referenciados pelas expressões a seguir.

mat	**5.8**	**7.9**	**4.5**	**3.8**	**1.2**	**2.3**	**7.9**	**4.6**	**2.9**	**2.35**	**8.9**	**10.9**

a) mat[2] b) mat[0] c) mat[13] d)mat[6]

2- Qual é a diferença entre os números “4” das duas instruções abaixo ?
int mat[4];
mat[4] = 5;

3- Quais serão os valores dos elementos do vetor x que serão impressos no final da execução do trecho de programa seguinte:

```
int main()
{
   int i,x[5];

for (i=0; i<=5; i++)
x[i] = 2;
for (i = 0; i<=5; i++)
x[i] = x[i]*i;
for (i=0;i<=5;i++)
printf("%d\n",x[i]);
   return 0;
}
```

4- Desenvolva um programa que leia um vetor de números reais, um escalar (também real) e imprima o resultado da multiplicação do vetor pelo escalar.

10-Matrizes(vetores multidimensionais):

Como os vetores, as matrizes são estruturas de dados homogêneas. Podem ser construídas dos diversos tipos básicos primitivos (float, int, char).

A Principal diferença em relação aos vetores (unidimensionais): possui uma ou mais dimensões adicionais,mas na maioria dos casos: utiliza-se matrizes bidimensionais.

São utilizadas quando os dados homogêneos necessitam de uma estruturação com mais de uma dimensão.

Exemplos:

Programar um jogo de xadrez (o tabuleiro é naturalmente bidimensional)**,**estrutura para guardar caracteres de um livro (três dimensões: 2 para representar os caracteres de uma página e uma terceira para indicar as páginas)**,**problemas matemáticos matriciais.

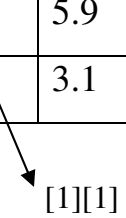
A sintaxe para declaração de uma variável deste tipo é semelhante a declaração dos vetores. Considera-se porém a quantidade de elementos da outra dimensão:
tipo_primitivo identificador[qde_el_linha] [qde_el_col];

Exemplo:
int mat[3][2]; {uma matriz de números inteiros com 3 linhas e 2 colunas, essa matriz possui 6 elementos}
Neste caso os índices variam de 0 a 2 para linhas e de 0 a 1 para colunas.

Caso geral:
tipo_primitivo identificador[dim 1] [dim 2]...[dim n];
matriz n-dimensional
Exemplo:
Float mat[2][3][4];

{matriz 2X3X4 do tipo float,esta matriz possui 24 elementos e os índices variam de 0 a 1 para a primeira dimensão, 0 a 2 para a segunda dimensão e 0 a 3 para a terceira dimensão}

	0	**1**	**2**	**3**
0	5.6	7.8	8.9	7.9
1	1.2	4.9	5.9	9.7
2	4.7	2.4	3.1	6.8



Pode-se fornecer valores de cada elemento de uma matriz na declaração, da mesma forma que nos vetores.
Exemplo:
float num[2][4] = {{3.6, 2.7},{5.0,4.1},{7.9,9.8}{5.3,2.4},{7.4,2.1}};

Ou seja, fornece-se os valores linha a linha.

Observações:

As matrizes não são inicializadas automaticamente.

Se houver menos valores do que o número de elementos da matriz, os elementos restantes são inicializados automaticamente com o valor zero.

int num[5][3] = {{32, 64, 27}};
int n[4][4] = {{0}}; //todos elementos são nulos.

A seguinte declaração causa um erro de sintaxe:

int n[2][5] = {{32, 64, 27, 18, 95, 14},{12,15,43,17,67,31}};

Uma vez que há mais elementos do que espaços de memória alocados.

Os elementos das matrizes são referenciados individualmente por meio de índices (iniciando de zero) entre colchetes

Exemplos:

mat

3	8	5
9	2	1

Dada a matriz acima tem-se:

A = mat[1][2];

B = mat[0][0];

*A instrução abaixo atribui um valor ao elemento linha zero e coluna um da matriz **mat**:*

i=0; j=1

mat[0][1] = 15; ou valores[i][j]=15;

Exercícios:

1- Quais são os elementos do vetor referenciados pelas expressões abaixo?

mat

3.2	4.1	2.7
5.9	0.6	9.0

a) mat[2][0] b) mat[1][1] c) mat[3][1]

2- Qual é a diferença entre os números "3" das duas instruções abaixo ?

int mat[6][3];

mat[6][3] = 5;

3- A instrução seguinte é correta para inicializar uma matriz ?

int mat[2][2] = {1, 2},{3,4};

4- Quais serão os valores dos elementos da matriz x no final da execução do trecho de programa seguinte:

```
for (i=0; i<=3; i++)
  for (j = 0; j<=3; j++)
    x[i][j]=i*(j+1);
for (i = 0; i<=3; i++)
  x[i][2] = x[2][i];
```

5- Desenvolva um programa para ler e somar duas matrizes quadradas de dimensão 2. Imprima o resultado da soma.

11-Vetores de Caracteres(Cadeias de caracteres ou *string*):

11.1-Caracteres em C:

Uma cadeia de caracteres é uma sequência de caracteres (*char*) justapostos e são fundamentais no desenvolvimento de programas computacionais.

Exemplos de cadeias de caracteres (representadas internamente num programa):

Mensagem de e-mail;

Texto de um programa;

Nome e endereço em cadastro de clientes, alunos, etc...

Sequência genética. Um gene (ou o DNA de algum organismo) é composto de sequências dos caracteres A, T, G e C (nucleotídeos).

Entrada/Saída de caracteres em C:

ch = getchar();

armazena **um** caractere digitado em **ch** até que ENTER seja pressionado;

putchar(ch);

Imprime o caractere armazenado na variável ch na tela do computador

Exemplo:

```
int main()
{
        char ch;
        ch = getchar();
        putchar(ch);
  return 0;
}
```

Observação: Por mais que se escreva uma frase extensa só será imprimido a primeira letra digitada.

Exemplo:

Um programa que peça para que você digite uma letra ou um número e informa se o que fora digitado fora ou não um número.

```
int main( )
{
        char c;
        printf("Digite uma letra ou numero: ");
        c = getchar();
        if((c >= '0') && (c<='9'))
                printf("Eh um numero");
        else
                printf("Nao eh um numero");
  return 0;


}
```

11.2-Cadeias de caracteres (strings):

Cadeias de caracteres, em C, são representadas por vetores do tipo *char* terminadas, **obrigatoriamente**, pelo caractere nulo: '\0' (\zero). Portanto, deve-se reservar uma posição para este caractere de fim de cadeia.

Exemplos:

```
char cidade[4] = {'R', 'i', 'o', '\0'};
char disc[40] = {'A','l','g', 'o', 'r', 'i', 't', 'm', 'o', '\0'};
```

Equivale:

```
char cidade[4] = "Rio";
char disc[40] = "Algoritmo";
```

Para ilustrar a declaração e a inicialização de *strings*, consideremos as seguintes declarações:

```
char s1[] = "" ; //2 aspas sem espaços entre elas
char s2[] = "Rio de Janeiro";
char s3[81];
char s4[81] = "Rio";
```

s1 armazena uma string vazia. Tem um único elemento: '\0';

s2 representa um vetor com 15 elementos (caracteres);(rio de janeiro contando os espaços 14 mais um para o '\0';

s3 representa uma cadeia de caracteres com até 80 caracteres e não é inicializada;(lembrando que a última posição é sempre do '\0';

s4 também é dimensionada para conter até 80 caracteres e é inicializada com a cadeia "Rio".

Leitura de cadeias de caracteres (scanf).

```
int main()
{
  char s[20];
        printf("Digite uma string: ");
        scanf("%s",s);  //sem & antes de s
        printf("String digitada: %s",s);
  return 0;
}
```

Lembrando que **//** insere um comentário sem ser impresso na tela ou alterar de alguma forma o algoritmo.

Neste caso, a leitura será feita **até** encontrar um caractere branco: espaço (' '), tabulação ('\t') ou nova linha ('\n'). Assim, se digitarmos "Rio de Janeiro", *s* conterá apenas "Rio"; Não é **necessário** o **&** antes da variável **s** em **scanf**.

11.3-Leitura/Impressão de cadeias de caracteres (gets/puts).

```
int main()
{
        char s[20];
  printf("Digite uma string: ");
        gets(s);
        printf("String digitada: ");
        puts(s);
  return 0;
}
```

Neste caso, se digitarmos “Rio de Janeiro“, *s* conterá “Rio de Janeiro“;
gets(s): lê a string *s* a partir do teclado;
puts(s): imprime uma string na tela seguida de nova linha.

Uma outra forma de imprimir uma cadeia de caracteres, caractere por caractere:

```
int main()
{
        char s[30]; //o tamanho poderia ser outro
        int i;
        printf("Digite uma string: ");
        gets(s);
        //imprime cada caractere da cadeia lida
        for(i=0; s[i]!='\0'; i++)
                printf("%c",s[i]);
  return 0;
}
```

* O **for** acima equivale a printf("%s",s);

Exemplo: o programa a seguir calcula e imprime o comprimento (numero de caracteres) de uma cadeia:

```
int main()
{
    char s[30];
    int i, n = 0;
    printf("Digite uma string: ");
    gets(s);

    for(i=0; s[i]!='\0'; i++)
      n++;
   printf("\n O tamanho de \"%s\" eh: %d",s,n);
   return 0;
}
```

Exemplo: O programa a seguir faz uma cópia de uma cadeia, fornecida pelo usuário, para outra:

```
int main()
{
        char dest[50],  //string destino
      orig[50]; //string origem
  int i;
  printf("Digite uma string: ");
        gets(orig);
  //copia cada caractere de orig para dest
        for(i=0; orig[i]!='\0'; i++)
                dest[i] = orig[i];
  //coloca o caractere nulo para marcar o fim da string
  dest[i] = '\0';
  puts(dest);
  return 0;
}
```

Exercícios:

1-Fazer um programa que leia o nome do usuário e o número de matricula e imprima.

12-Bibliografia:

- Servidor de vídeo RIO: www.land.ufrj.br/rio. Network Simulator (NS2).

- Introdução à linguagem versão 2.0 Gerência de Atendimento ao Cliente Centro De Computação UNICAMP

-Slides disponíveis no site sites.google.com/site/rodrigoluis/cursos