

# Algoritmos e Programação II

- Laços
- Array (vetor)
- Matrizes

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

# Bibliografia



**Título:** Como Programa C – Sexta Edição

**Editora:** Pearson

**Autor:** Paul Deitel

# Em um momento no Passado...

IF...

IF.. ELSE...

IF...ELSE...IF

Operadores: + - \* / ^

Prioridade do Operadores: ()\* / + - ^

Operadores Lógicos: AND e OR



INSTITUTO FEDERAL  
RIO GRANDE DO SUL

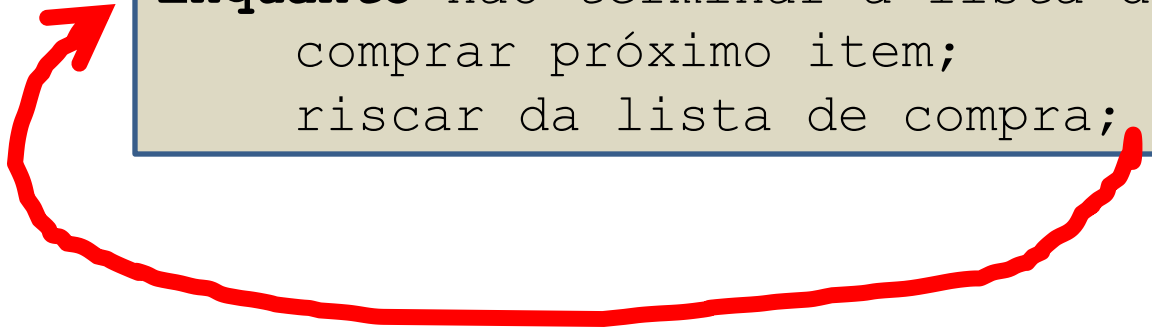
# O que é um Algoritmo?



# Repetição (Laços)

Uma estrutura de repetição possibilita a repetição de uma ação enquanto uma determinada condição não for satisfeita.

Exemplo (portugol)



```
Enquanto não terminar a lista de compras  
    comprar próximo item;  
    riscar da lista de compra;
```

# Repetição Indeterminada: WHILE



# Repetição (WHILE)

Exemplo (estrutura C)

```
.  
.   
.   
produto = 3;  
while (produto <= 100)  
{  
    produto = 3 * produto;  
    printf ("valor de Produto: %d" \n, produto);  
}  
.   
.   
. 
```

# Repetição (WHILE)

- Em uma estrutura de repetição do Tipo **While**, existe um tipo de variável responsável pelo controle do laço. Essa variável é conhecida como **contador**.
- Sem o contador, o laço se torna um “laço infinito”.
- Nesse tipo de laço **While**, a condição é checada no início do laço: ou seja, pode executar nenhuma vez.



# Repetição (**WHILE**)

A instrução **while** executa uma instrução ou um bloco de instruções até que uma expressão especificada seja avaliada como *false*, ou seja, executa um bloco enquanto a condição for verdadeira.

A instrução **while** é usada para repetições indeterminadas, e para cada repetição é avaliada uma nova condição.

Se na condição avaliada o resultado for verdadeiro, será executado mais uma vez o bloco.

# Repetição (WHILE)

O controle de laço **WHILE** é manual, o seja, o contador deve ser descrito dentro do bloco **while** juntamente com o seu incremento.

```
int i = 0;
While (i<= 10)
{
    ...
    i =i + 1;
    ...
}
```

O while executa o **teste no inicio do bloco**. Se este teste obtiver o resultado verdadeiro, executará uma vez o bloco.

# Repetição (While)

```
main()  
{  
    int n = 1;  
    while (n < 6)  
    {  
        printf("Esta é a linha %d", n);  
        n++;  
    }  
}
```

- Neste caso, o bloco de operações será executado enquanto a condição **(n < 6)** for verdadeira.
- O teste da condição será sempre realizado antes de qualquer operação (teste no início).

# Repetição (Do... While)

```
main()  
{  
    int n = 1;  
    do  
    {  
        printf("Esta é a linha %d", n);  
        n++;  
    } while (n<6);  
}
```

- Neste caso primeiro são executados os comandos, e somente depois é realizado o teste da condição (**teste no final**).
- Se a condição for verdadeira, os comandos são executados novamente, caso seja falso é encerrado o comando **DO..**

# Repetição (While)

## Dicas:

- Utilizar o comando em caixa baixa : **while**.
- O comando **while** não tem ponto e virgula no final da linha.
- As instruções devem estar demarcadas no bloco de repetição através de { }.
- Todas as instruções dentro do bloco **while** devem seguir as regras do ponto e vírgula.
- O contador do bloco de repetição é incrementado manualmente.

# Repetição Determinada “FOR”

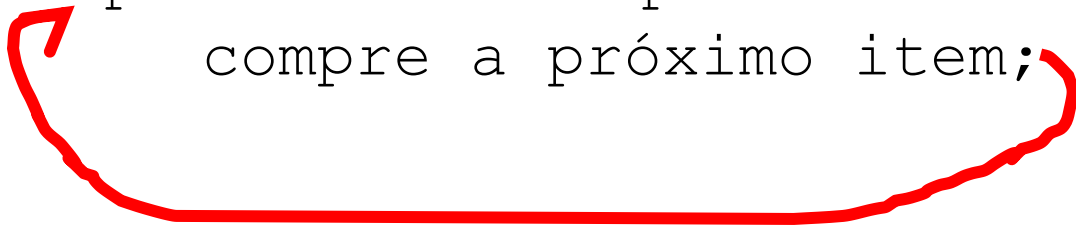


# Repetição FOR

- Neste tipo de repetição, uma ação é executada um determinado número de vezes (definido o valor inicial, valor final e incremento).
- O contador, que controla a quantidade de execuções de um bloco, é incrementado automaticamente.

Exemplo (portugol)

```
para lista completa  
    compre a próximo item;
```



# Repetição (FOR)

Exemplo (estrutura C)

```
.  
.   
.   
for(int produto = 3; produto <= 100; produto=produto *3)  
{  
    printf ("valor de Produto: %d" \n, produto);  
}  
.   
.   
. 
```



# Repetição (FOR)

Quando existe a necessidade de interromper o processamento antes de ser alcançado o limite final definido para a estrutura, deverá ser usado o comando **break**.

```
main()  
{  
  for (int cont = 1; cont <= 10; cont++)  
  {  
    printf("% d", cont);  
    if (cont == 5)  
      break;  
  }  
}
```



# Operadores

## Operadores de Atribuição:

<code>+=</code>	<code>c += 7</code>	<code>c = c + 7</code>
<code>-=</code>	<code>c -= 6</code>	<code>c = c - 6</code>
<code>*=</code>	<code>c *= 3</code>	<code>c = c * 3</code>
<code>/=</code>	<code>c /= 4</code>	<code>c = c / 4</code>
<code>%=</code>	<code>c %= 2</code>	<code>c = c % 2</code>

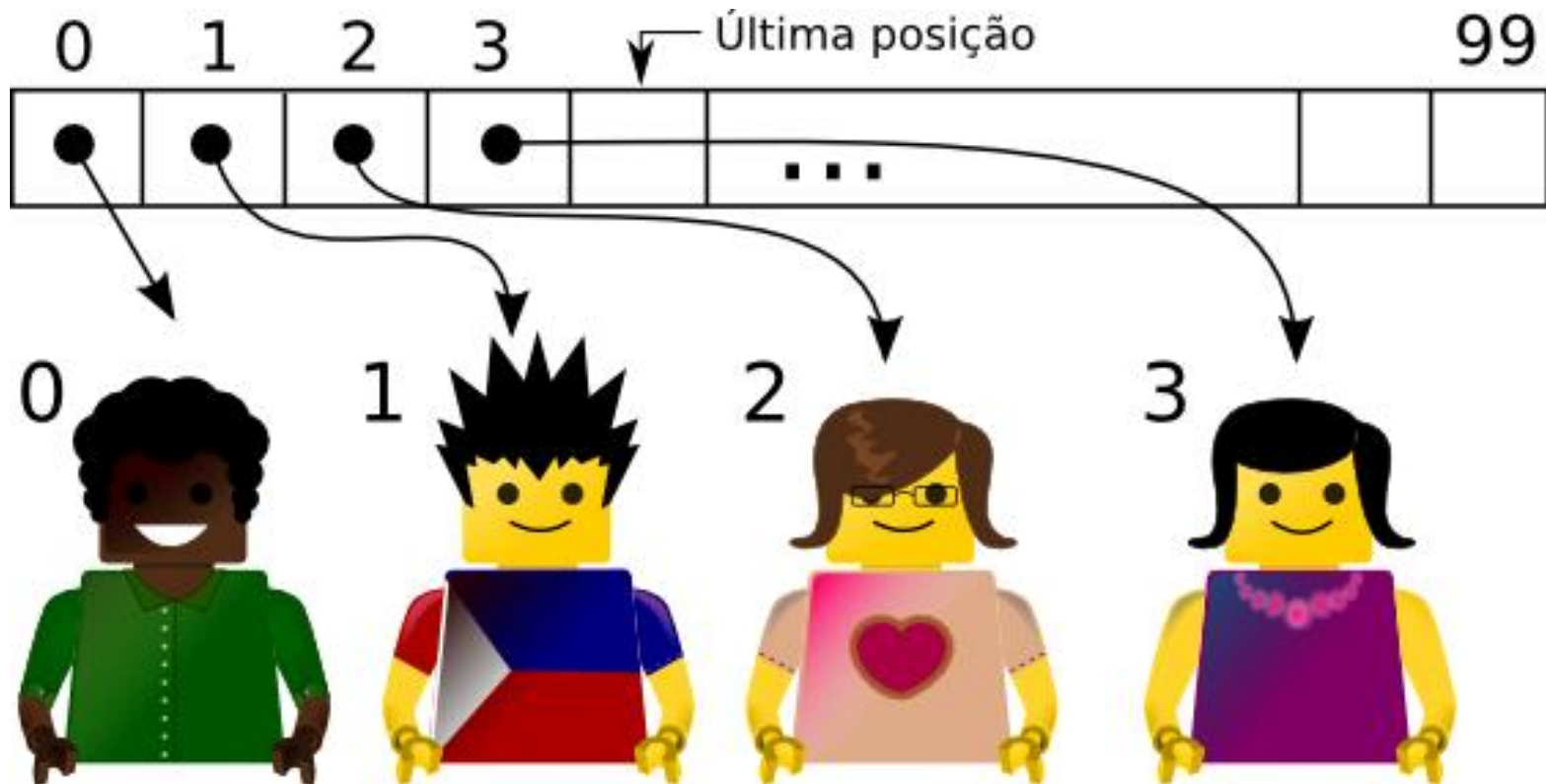
## Operadores de Incremento/Decremento:

<code>++</code>	<code>a++</code>	primeiro incremento, depois avalia;
<code>++</code>	<code>++a</code>	primeiro avalia, depois incrementa
<code>--</code>	<code>a--</code>	primeiro decrementa, depois avalia;
<code>--</code>	<code>--a</code>	primeiro avalia, depois decrementa

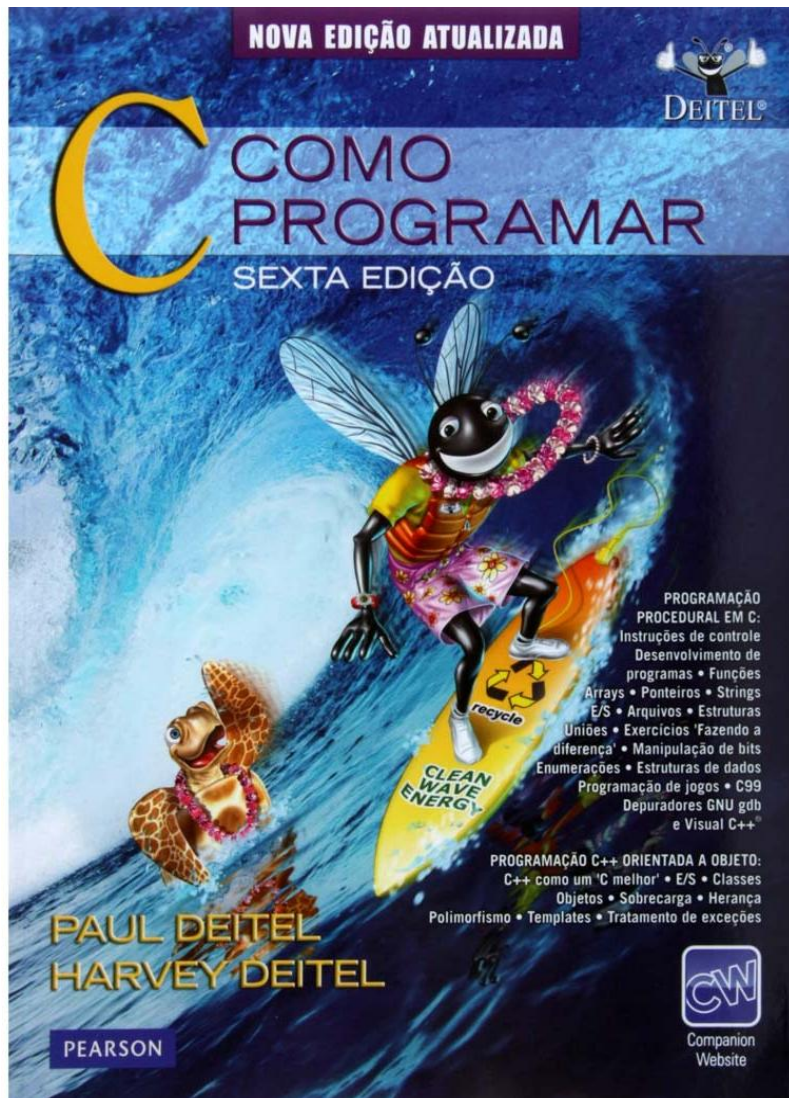


INSTITUTO FEDERAL  
RIO GRANDE DO SUL

# Vetores (array)



# Bibliografia Básica



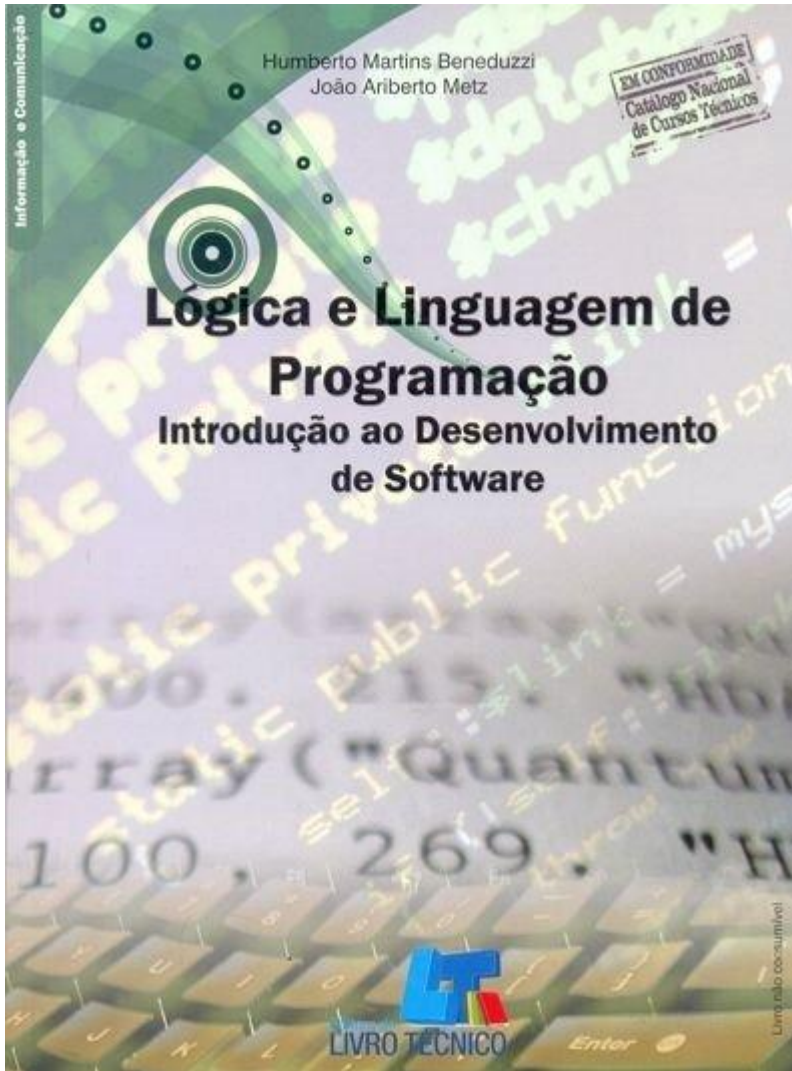
Capítulo 6 (pag 242)

Arrays



INSTITUTO FEDERAL  
RIO GRANDE DO SUL

# Bibliografia Complementar



**Capítulo 5 (pag 59)**

**Estruturas de Dados Homogêneas**

# Vetor

Pode ser definido com uma variável **multivalorada**, ou seja, uma variável que pode armazenar múltiplos valores indexados e do mesmo tipo.

É uma estrutura multivalorada **homogênea** com acesso aos dados através de um índice.

## Exemplo

```
ListaProdutos[0]= sabao  
ListaProdutos[1]= leite  
ListaProdutos[2]= cafe  
ListaProdutos[3]= arroz
```

# Vetor (criação)

Para criar um vetor é necessário definir o tipo de dado e a quantidade de posições reservadas ao vetor.

Um vetor em “C” sempre é iniciado na posição 0 (zero).

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

main()
{
    int valor[3];
    printf ("\nInforme o primeiro valor: ");
    scanf ("%d", &valor[0]);
    printf ("\nInforme o segundo valor: ");
    scanf ("%d", &valor[1]);
    printf ("\nInforme o terceiro valor: ");
    scanf ("%d", &valor[2]);
    printf ("\nVoce digitou os valores: %d, %d e %d", valor[0], valor[1], valor[2]);
    system("pause");
}
```

# Vetor (manipulação)

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

```
int main ()
{
    int num[100];
    int count=0;
    int totalnums;
```

```
do
{
    printf ("\nEntre com um numero (-999 p/ terminar): ");
    scanf ("%d",&num[count]);
    count++;
} while (num[count-1] != -999);
```

```
totalnums=count-1;
```

```
printf ("\n\n\n\t Os numeros digitados foram:\n\n");
```

```
for (count=0;count < totalnums;count++)
    printf (" %d",num[count]);
return(0);
}
```



# Vetores

## Dicas:

- Considerando a declaração:

**int valor[10];**

o “C” reserva espaço de memória **contígua** iniciado da posição 0 até a posição 9.

- A estrutura C não verifica se o índice usado está dentro dos limites válidos, ou seja, é possível usar **valor[34]**.

# Vetores do Tipo String

- Uma string é um tipo de variável especial, definida como um vetor por definição.
- O marcador de final de string é o “\0”.

Exemplo:

```
char word[10] = {“cafe” };
```

```
Char[0] = “c”
```

```
Char[1] = “a”
```

```
Char[2] = “f”
```

```
Char[3] = “e”
```

```
Char[4] = “\0”
```

# Vetores do Tipo String

```
#include<stdio.h>
#include<stdlib.h>
main()
{
    char palavra[10] ;
    int i=0;

    printf("Digite uma Palavra ateh 10 caracteres: ");
    scanf("%s",&palavra);

    while (i< 9)
    {
        printf ("\nA letra %d eh: %c", i, palavra[i]);
        i++;
    }

    system("pause");
}
```

# Vetores

## Dicas:

- Para localizar o final de uma palavra basta localizar '\0'

```
if (palavra[i]=='\0')  
    break;
```

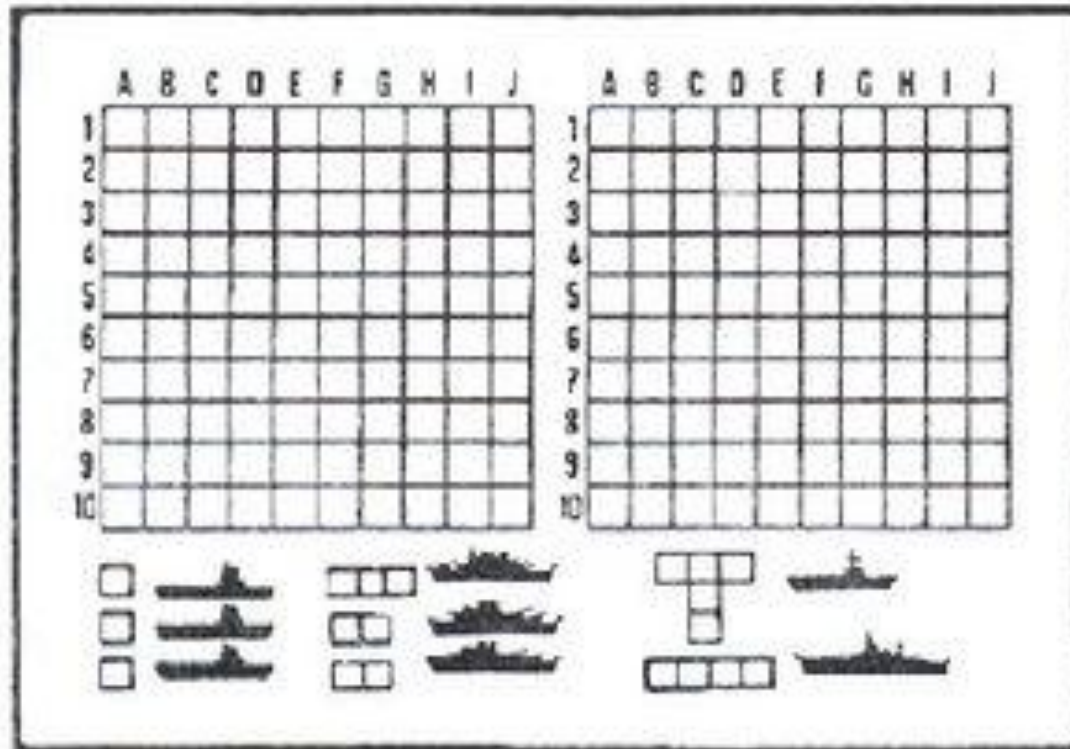
- Para criar um vetor e atribuir valores:

```
int numero[7] = {1,3,5,7,8,3,4};  
int palavra[7] = "cafe\0";
```

- Para ler uma frase:

```
scanf ("%[A-Za-z 0-9]s", &palavra);
```

# Matrizes



# Matriz

Uma matriz pode ser considerado um vetor (array) de duas dimensões, também chamada vetor bi-dimensional,

Seu funcionamento é praticamente da mesma forma que um vetor, exceto que é utilizado um número  $n$  de índices para acessar uma dimensão.

Nesta estrutura, o índice da esquerda indexa as linhas e da direita indexa as colunas.

**Na linguagem C, os índices variam de zero ao valor declarado (menos um);**

**A linguagem C não verifica se o intervalo dos índices estão na faixa declarada. Manter os índices na faixa permitida é tarefa do programador.**



# Matriz (definição)

A declaração de uma matriz em português estruturado pode ser da seguinte forma:

```
tipo Nome_Matriz [tamanho_linha][tamanho_coluna]
```

## Exemplo de declaração em C

```
int matriz [10][10] ;
```

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

```
float media [3][3];
```

0	1	2
1		
2		

# Matriz (inicialização)

Exemplo de inicialização de matrizes multidimensionais, onde `matrix` está sendo inicializada com 1, 2, 3 e 4 em sua primeira linha, 5, 6, 7 e 8 na segunda linha e 9, 10, 11 e 12 na última linha.

```
int matrix [3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
```

Inicialização de um vetor de strings multidimensional:

```
char str_vect [3][10] = { "Joao", "Maria", "Jose" };
```



# Matriz (manipulação)

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

```
main()
```

```
{
```

```
    //DECLARAÇÃO DE VARIÁVEIS - ESCOPO LOCAL
```

```
    int soma, i, j, matriz[2][2];
```

```
    matriz[0][0] = 0;
```

```
    matriz[0][1] = 1;
```

```
    matriz[1][0] = 2;
```

```
    matriz[1][1] = 3;
```

```
    printf("Linha 1 coluna 1 %d \n", matriz[0][0]);
```

```
    printf("Linha 1 coluna 2 %d \n", matriz[0][1]);
```

```
    printf("Linha 2 coluna 1 %d \n", matriz[1][0]);
```

```
    printf("Linha 2 coluna 2 %d \n", matriz[1][1]);
```

```
    system("pause");
```

```
}
```

0	1
2	3



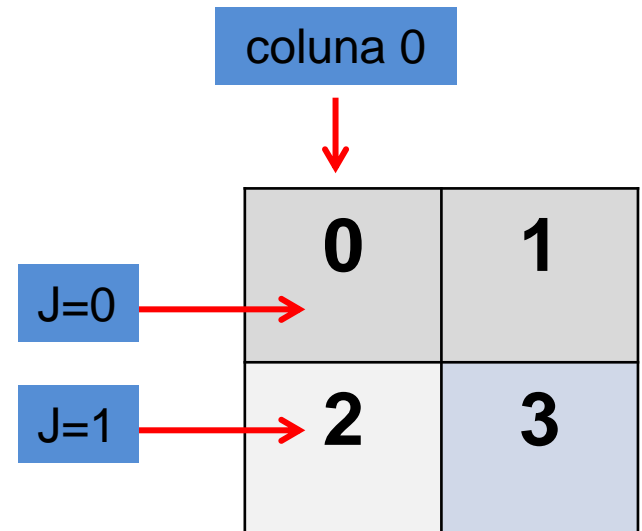
INSTITUTO FEDERAL  
RIO GRANDE DO SUL

# Matriz (manipulação com Laço)

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

main()
{
    //DECLARAÇÃO DE VARIÁVEIS - ESCOPO LOCAL
    int soma, i, j, matriz[2][2];
    matriz[0][0] = 0;
    matriz[0][1] = 1;
    matriz[1][0] = 2;
    matriz[1][1] = 3;

    for (j=0; j<2; ++j) // Laço para as Linhas
    {
        printf("Linha j coluna 0 %d \n", matriz[j][0]);
    }
    system("pause");
}
```





INSTITUTO FEDERAL  
RIO GRANDE DO SUL

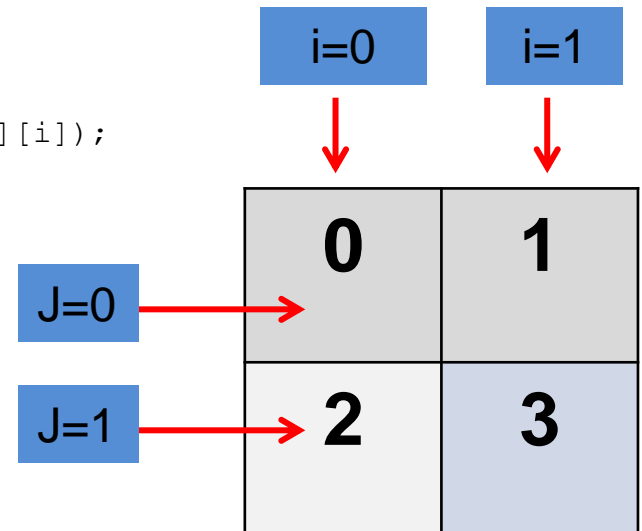
# Matriz (manipulação com Laço)

```
#include<stdio.h>

#include<stdlib.h>

main()
{
    //DECLARAÇÃO DE VARIÁVEIS - ESCOPO LOCAL
    int soma, i, j, matriz[2][2];
    matriz[0][0] = 0;
    matriz[0][1] = 1;
    matriz[1][0] = 2;
    matriz[1][1] = 3;

    for (j=0; j<2; ++j) // Laço para as Linhas
    {
        for (i=0; i<2; ++i) //Laco para as colunas
        {
            printf("Linha j coluna i %d \n", matriz[j][i]);
        }
    }
    system("pause");
}
```



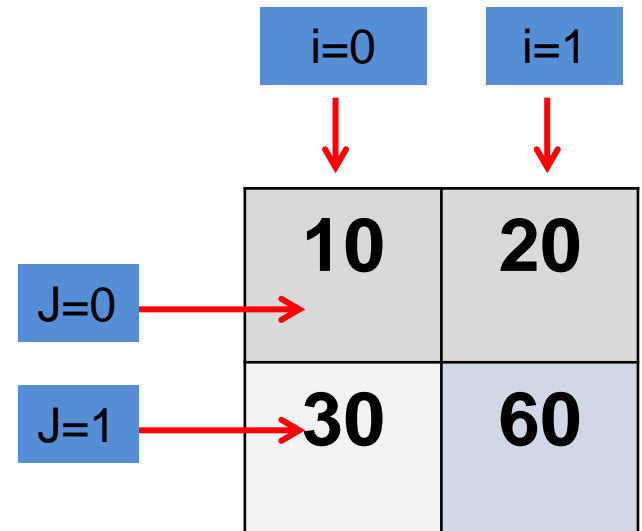


INSTITUTO FEDERAL  
RIO GRANDE DO SUL

# Matriz (manipulação com Laço)

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

main()
{
    //DECLARAÇÃO DE VARIÁVEIS - ESCOPO LOCAL
    int soma, i,j;
    int matriz2[2][2]={10,20,30,60};
    //Somar todos os valores
    soma = 0;
    for (j=0; j<2; ++j) // Laço para as Linhas
    {
        for (i=0; i<2; ++i) //Laco para as colunas
        {
            soma = soma + matriz2[j][i];
        }
    }
    printf("A soma eh %d \n", soma);
    system("pause");
}
```





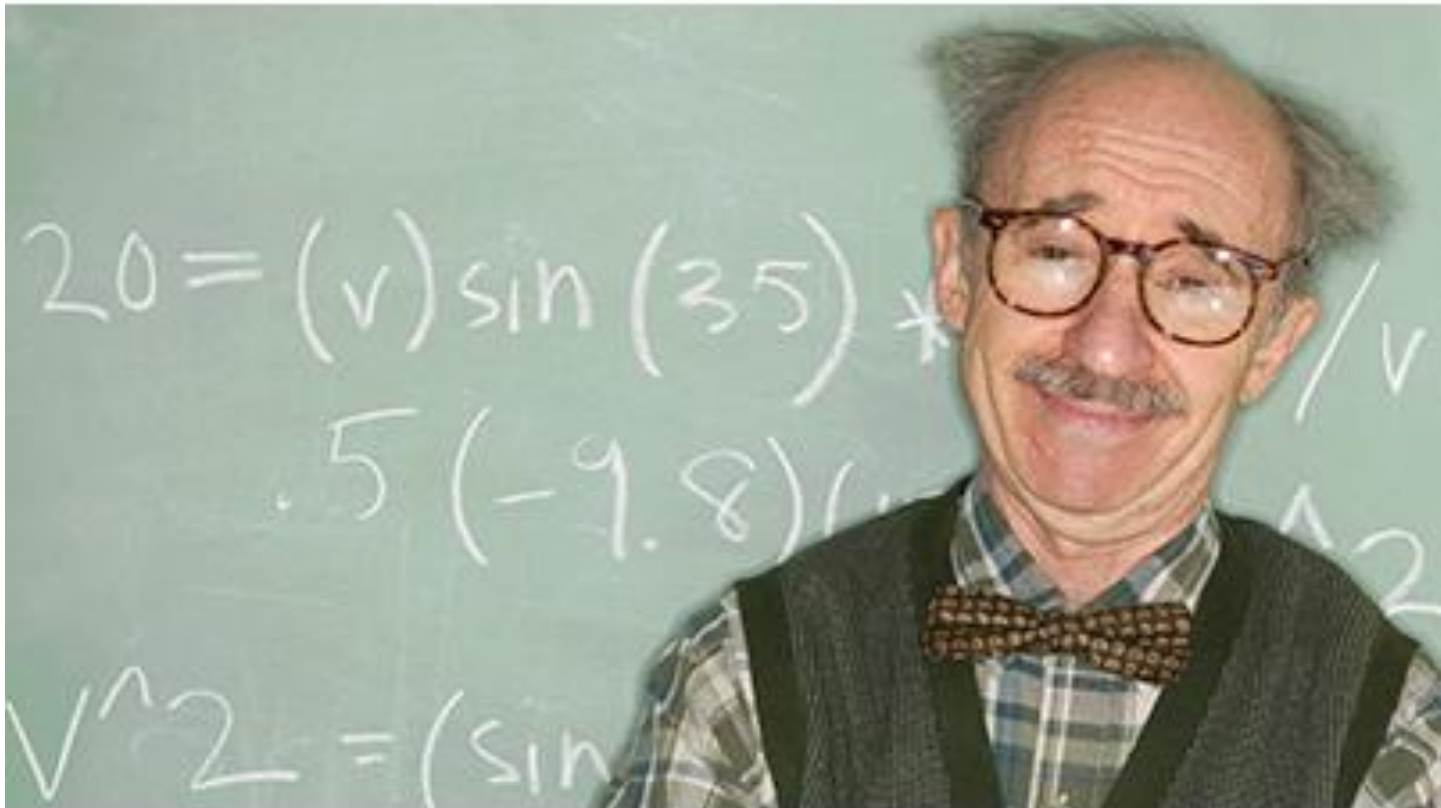
INSTITUTO FEDERAL  
RIO GRANDE DO SUL

# Matriz (manipulação com Laço)

```
#include<stdio.h>
#include<stdlib.h>
#define t 3
int main(){

    int mat[t][t], i, j;
    for(i = 0; i<t; i++)
    {
        for(j = 0; j<t; j++)
        {
            printf("Digite o elemento mat[%d][%d]", i, j);
            scanf("%d", &mat[i][j]);
        }
    }
    for(i = 0; i<t; i++)
    {
        for(j=0; j<t; j++)
        {
            printf("mat[%d][%d] = %d, ", i, j, mat[i][j]);
        }
        printf("\n");
    }
    system("pause");
    return 0;
}
```

# Exercícios



# While – Exercícios

- 1) Faça um programa de conte de 10 até 20;
- 2) Faça um programa que conte de 10 até 20, mostrando somente os números pares;
- 3) Faça um programa que leia um valor inicial e um valor final e mostre todos os valores do intervalo.
- 4) Faça um programa que leia valores inteiros e mostre enquanto não informado o valor 0 (zero).



# FOR- Exercícios

- 5) Faça um programa de conte de 10 até 20;
- 6) Faça um programa que conte de 10 até 20, mostrando somente os números pares;
- 7) Faça um programa que leia um valor inicial e um valor final e mostre todos os valores do intervalo.
- 8) Faça um programa que leia valores inteiros e mostre enquanto não informado o valor 0 (zero).





# Exercícios

9) Fazer uma programa que leia uma frase. Após ler a frase, deverá colocar todas as vogais em maiúsculas.

Fazer dois programas, um utilizando o laço FOR (9.a) e outro WHILE (9.b).

10) Fazer programa que leia o Peso e a Identificação de 20 Pessoas. Após a leitura dos 20 critérios, mostrar:

- a) O mais pesado;
- b) O menos pesado;
- c) A soma dos pesos;
- d) A média dos pesos;



# Exercícios

11) Fazer programa que leia o **Peso** e a **Identificação** de um grupo indeterminado de pessoas (até que seja informado o valor -1)

Após a leitura dos critérios mostrar:

- a) O mais pesado;
- b) O menos pesado;
- c) A soma dos pesos;
- d) A média dos pesos;



# Exercícios

12) Faça um programa que leia o nome / sobrenome, a Idade e sexo de um grupo não determinado de pessoas, - enquanto for respondido “S” no termino do cadastro.

**Após a leitura deverá mostrar:**

- a) A média de Idade dos Homens;
- b) A quantidade de mulheres;
- c) O nome da pessoa mais **idosa**;
- d) O nome da última mulher cadastrada.





# Exercícios

13) Considerando que a tabela a seguir e o tempo (horas) que um avião leva entre as cidades:

	0	1	2	3	4	5	6
0	0	02	11	06	15	11	01
1	02	0	07	12	04	02	15
2	11	07	0	11	08	03	13
3	06	12	11	0	10	02	01
4	15	04	08	10	0	05	13
5	11	02	03	02	05	0	14
6	01	15	13	01	13	14	0

Fazer um programa que mostre o tempo necessário para percorrer duas cidades fornecidas, até o momento em que fornecer duas cidades iguais.



# Exercícios

- 14) Considerando a tabela abaixo, fazer um programa que:
- a) Inverta os valores da primeira Coluna com a última (mostrar a matriz);
  - b) Inverta os valores da primeira linha com a última. (mostrar);
  - c) Mostrar a soma das linhas e colunas.
  - d) Mostrar a soma da matriz.

	0	1	2	3	4	5	6
0	02	02	11	06	15	11	01
1	02	14	07	12	04	02	15
2	11	07	13	11	08	03	13
3	06	12	11	13	10	02	01
4	15	04	08	10	14	05	13
5	11	02	03	02	05	13	14
6	01	15	13	01	13	14	02



# Exercícios

- 16) Considerando as tabelas abaixo, fazer um programa que:
- a) Realize a soma das tabelas;
  - b) Faça a multiplicação das tabelas;

	0	1
0	03	04
1	02	06

	0	1
0	05	08
1	07	1



# Exercícios

17) Faça um algoritmo que preencha uma matriz da seguinte forma:

	0	1	2	3	4	5	6
0	X	-	-	-	-	-	-
1	-	X	-	-	-	-	-
2	-	-	X	-	-	-	-
3	-	-	-	X	-	-	-
4	-	-	-	-	X	-	-
5	-	-	-	-	-	X	-
6	-	-	-	-	-	-	X



# Exercícios

18) Faça um algoritmo que preencha uma matriz da seguinte forma:

	0	1	2	3	4	5	6
0	X	-	-	-	-	-	X
1	-	X	-	-	-	X	-
2	-	-	X	-	X	-	-
3	-	-	-	X	-	-	-
4	-	-	X	-	X	-	-
5	-	X	-	-	-	X	-
6	X	-	-	-	-	-	X





# Exercícios

19) Faça um algoritmo que preencha uma matriz da seguinte forma:

	0	1	2	3	4	5	6
0	X	X	X	X	X	X	X
1	-	X	X	X	X	X	X
2	-	-	X	X	X	X	X
3	-	-	-	X	X	X	X
4	-	-	-	-	X	X	X
5	-	-	-	-	-	X	X
6	-	-	-	-	-	-	X



# Exercícios

20) Faça um algoritmo que preencha uma matriz da seguinte forma:

	0	1	2	3	4	5	6
0	X	X	X	X	-	-	-
1	X	X	X	X	-	-	-
2	X	X	X	X	-	-	-
3	X	X	X	X	X	X	X
4	-	-	-	X	X	X	X
5	-	-	-	X	X	X	X
6	-	-	-	X	X	X	X