# Arranjo multidimensional

Roberto Rocha

PUC Minas Virtual

# Programar

PUC Minas Virtual

# Ser programador*

A matéria-prima de trabalho de um programador de computador, por mais estranho que parecia, é a logica de programação e não a linguagem de programação em si.

A linguagem de programação, seja qual for, é apenas uma ferramenta que possibilita concretizar na memoria de um computador digital o programa idealizado na mente do profissional de programação de computadores.

Por esta razão é ideal que esse profissional conheça algumas linguagens de programação; quanto mais linguagens conhecer e trabalhar ao mesmo tempo, melhor.

Muitos profissionais dessa área, por falta de uma devida orientação e educação técnica adequada, baseiam seu aprendizado de programação na ferramenta, em uma única linguagem de programação.

Há aqueles que focam o aprendizado em uma única linguagem e não no exercício de aplicação da logica de programação, que usa algoritmos e desenvolve projetos lógicos devidamente documentados.

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo de. **Algoritmos :** lógica para desenvolvimento de programação de computadores. 28. ed. rev. e atual. São Paulo, SP: Érica, 2016. 1 recurso online ISBN 9788536518657 pag.176

# Ser programador*

O problema dessa atitude é que uma linguagem de programação pode deixar de ser utilizada ou ter seu uso reduzido no mercado por qualquer motivo.

Neste caso, um programador pode não ser mais necessário nesse mercado.

Pelo fato de voltar o aprendizado à parte pratica de uma determinada linguagem, acaba por perder muito da sensibilidade algorítmica de que precisa para programar de forma mais ampla e também aprender novas linguagens.

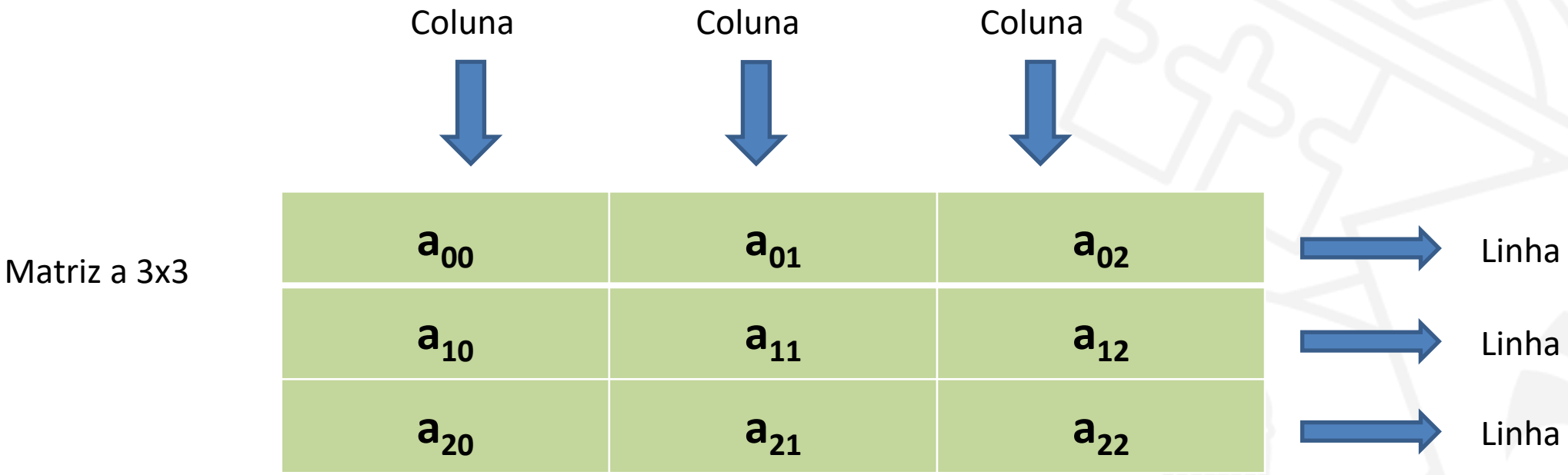
Quando um programador aprende a programar um computador sobre uma ferramenta de linguagem de programação, acaba por desenvolver uma mentalidade preconceituosa, achando que a linguagem de programação que ele "sabe" usar é a melhor e que outras linguagens desconhecidas por ele não prestam.

Cada linguagem de programação foi criada para solucionar categorias distintas de problemas, portanto uma linguagem de programação não é melhor que a outra. Muitas vezes as linguagens se complementam, tanto que é comum usar no desenvolvimento de um único sistema varias linguagens de programação. O maior exemplo dessa diversidade é a própria Internet, em que o desenvolvimento de um sitio se faz com varias linguagens, como XHTML, CSS, PHP, Ajax, entre outras.

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo de. **Algoritmos :** lógica para desenvolvimento de programação de computadores. 28. ed. rev. e atual. São Paulo, SP: Érica, 2016. 1 recurso online ISBN 9788536518657 pag.176

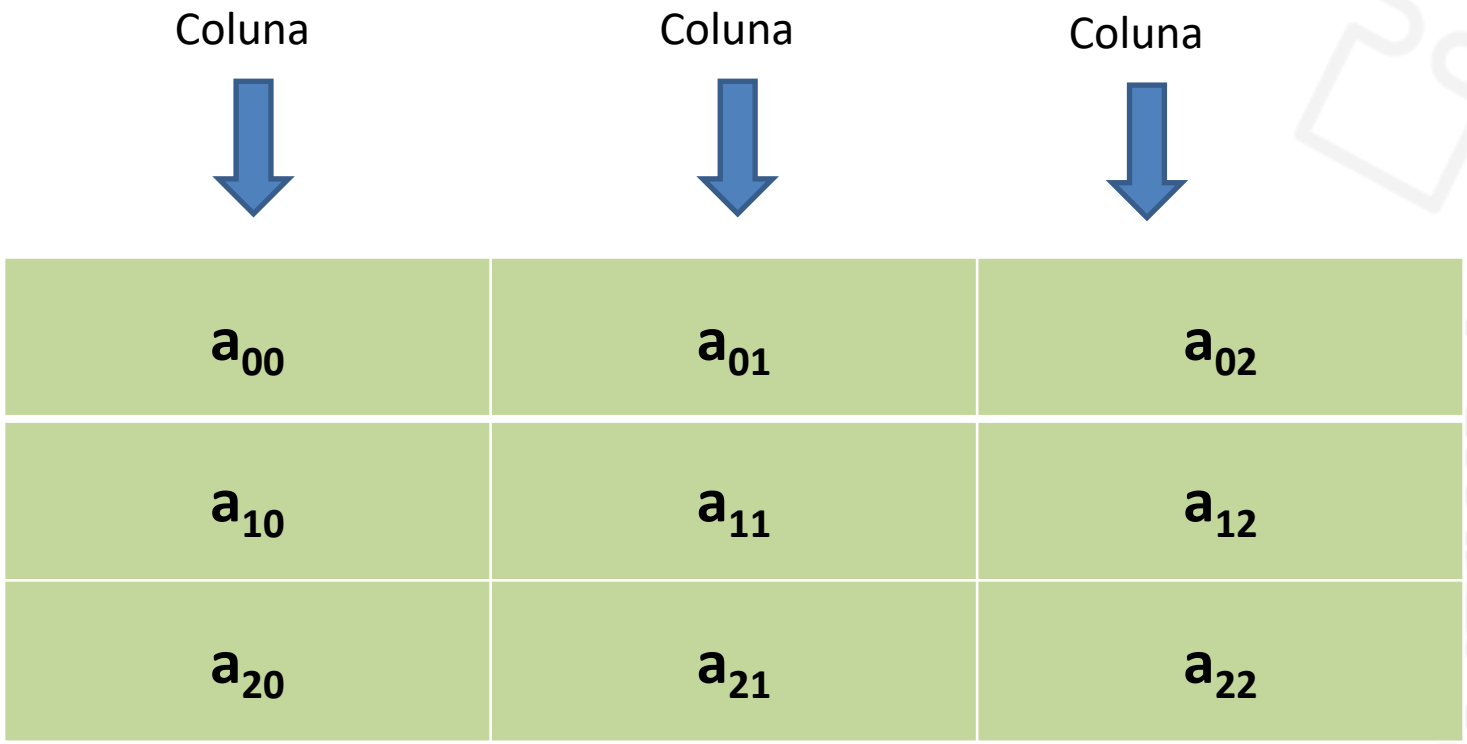
# Matrizes de Duas Dimensões

Uma matriz de duas dimensões faz menção a um elemento armazenado em uma linha e coluna.



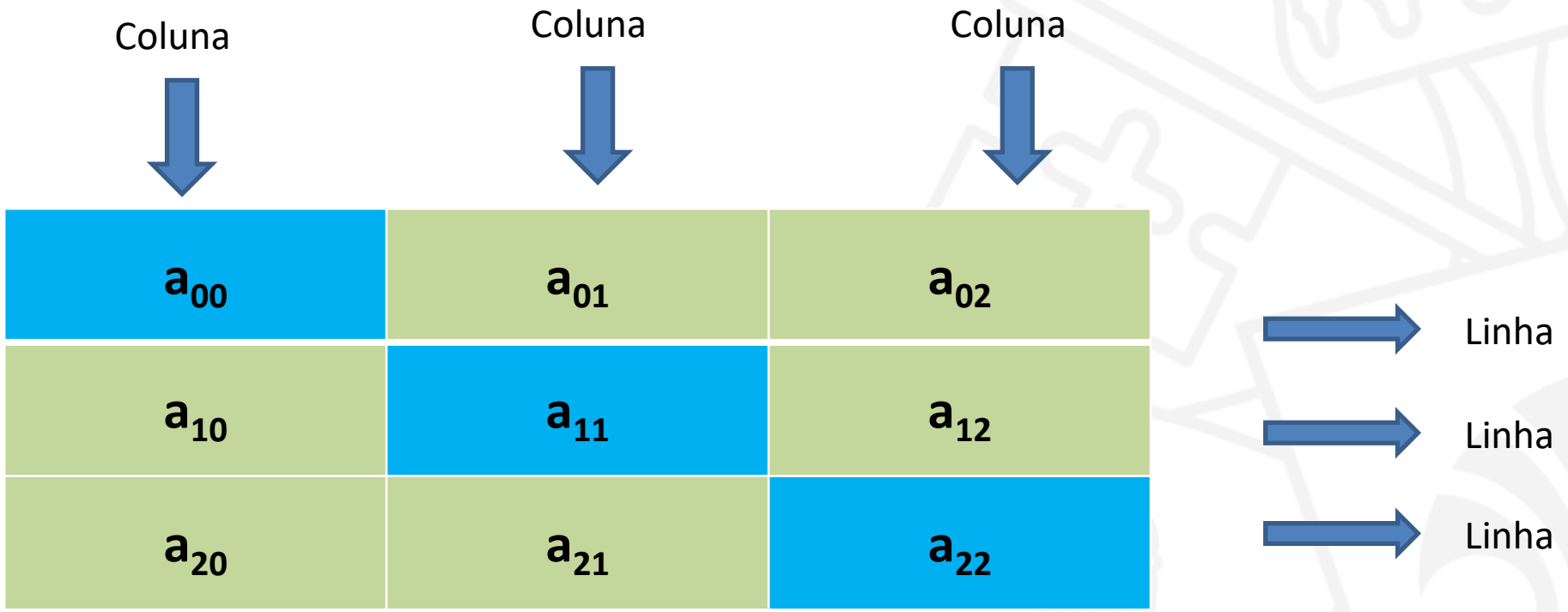

Este **tipo de variável** permite representar uma grande quantidade de dados, similares em tipo, **por um mesmo nome**. **Cada elemento,** em particular, é endereçado através de **índices** que **indicam uma posição** deste elemento no conjunto.

# Matrizes de Duas Dimensões



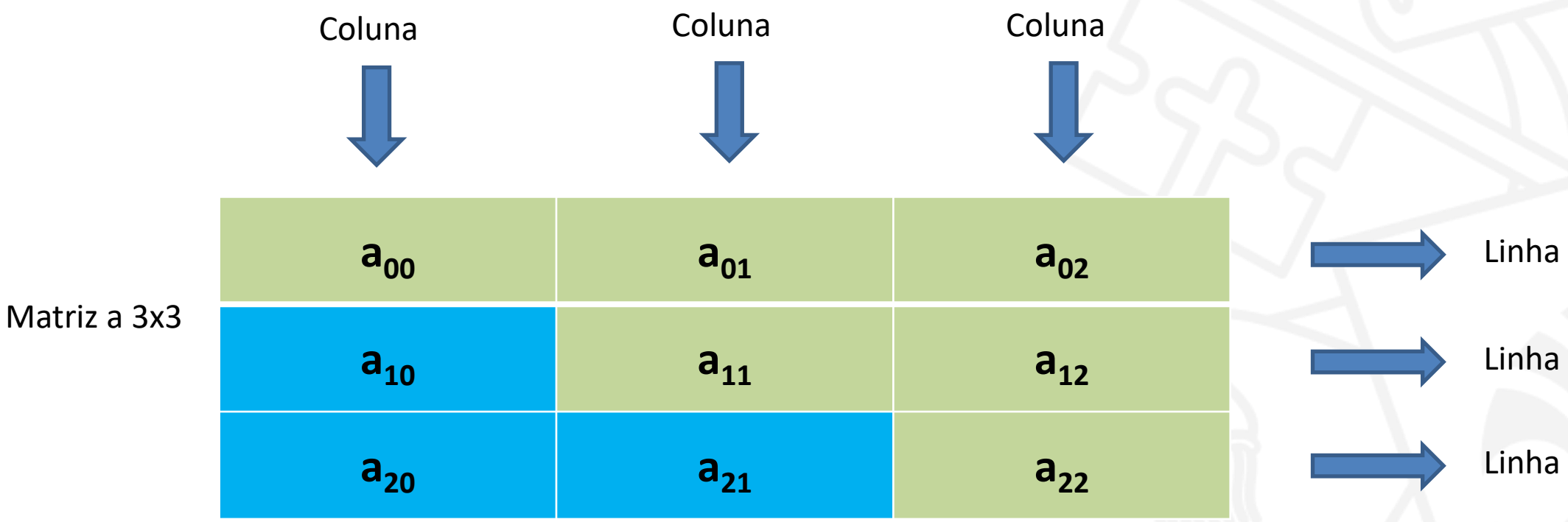
# Matrizes de Duas Dimensões

Matriz a 3x3



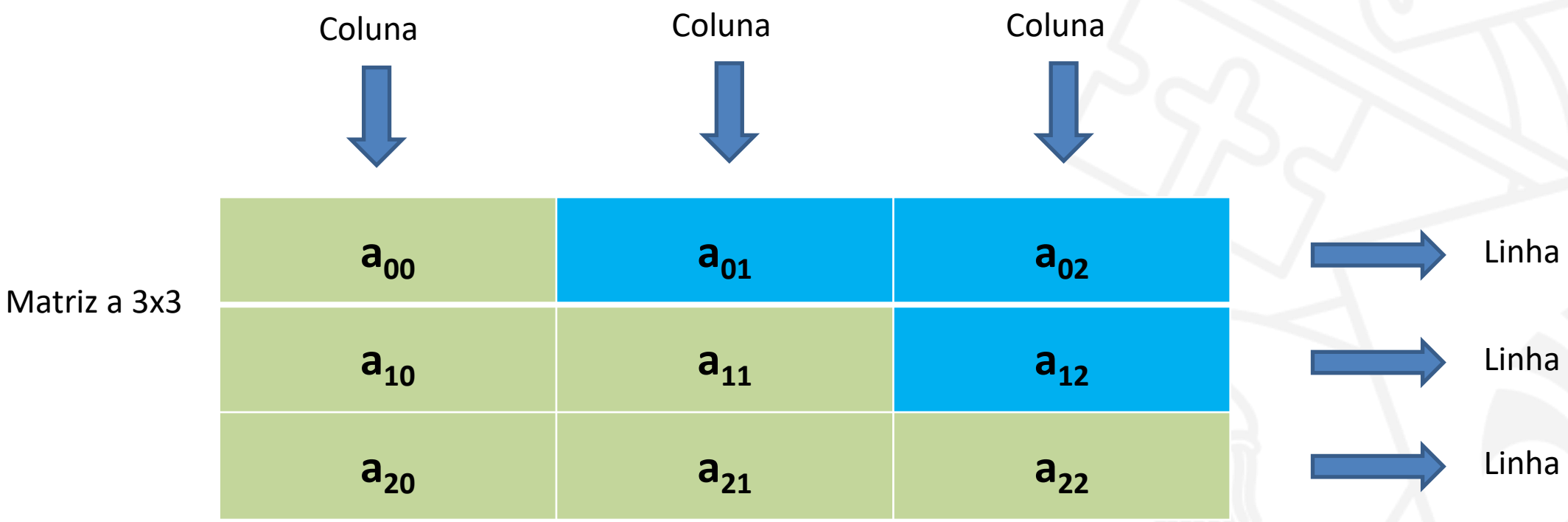

Diagonal principal: linha = coluna

# Matrizes de Duas Dimensões



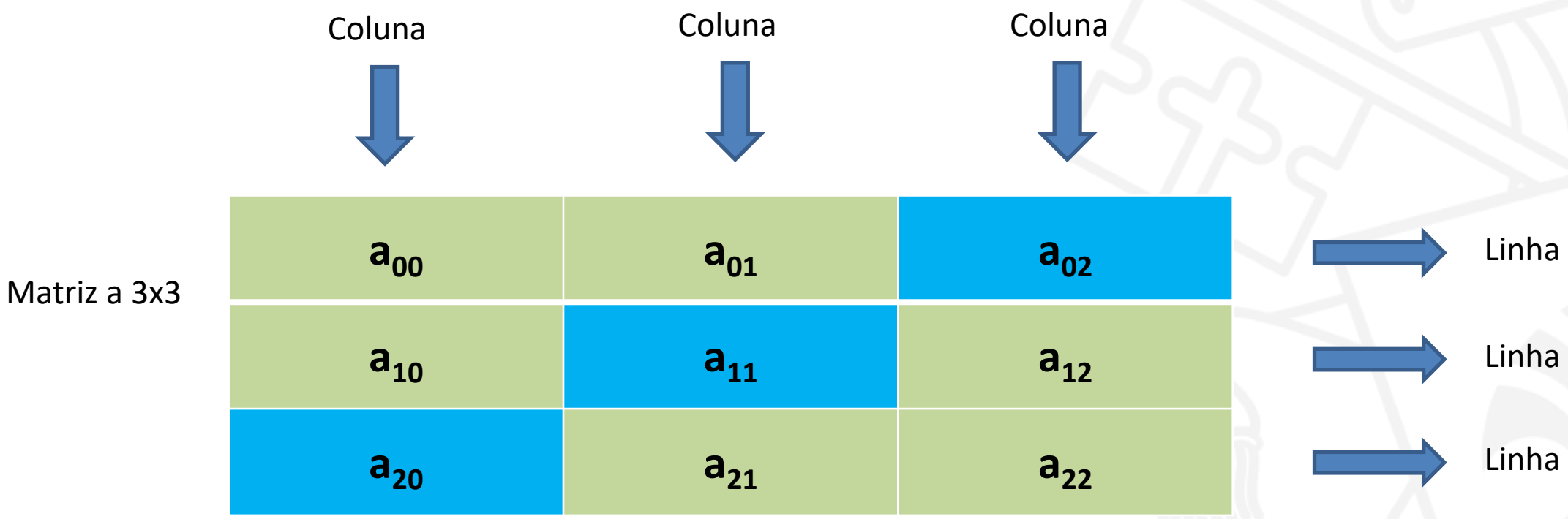

Elementos abaixo da diagonal principal: linha > coluna

# Matrizes de Duas Dimensões




Elementos acima da diagonal principal: linha < coluna

# Matrizes de Duas Dimensões




Elementos da diagonal secundária: linha + coluna = dimensão - 1

# Arranjo duas Dimensão - Matriz

**Definição de dados indexados:**

**A forma geral para se definir uma variável agrupada é a mesma de um variável simples, seguida pela definição de elementos do grupo, por dimensão.**

**Em Matlab o primeiro valor ocupará a posição de índice igual a um (1)**

**Em C o primeiro valor ocupará a posição de índice igual a zero(0)**

Em algoritmo, tanto <*valor-inicial*> como <*valor-final*> devem ser inteiros. Além disso, exige-se evidentemente que <*valor-final*> seja maior do que <*valor-inicial*>.

Exemplos:

```
vet: conjunto [0..4,0..4] de real   // reserva 25 posições de memória variável vet
qtd: conjunto [0..9,0..9] de inteiro // vetor de 100 posições para qtd
```

Em C

```
float vet[5][5];   // reserva 25 posições de memória variável vet 5 linhas [0 a 4] e 5 colunas [0 a 4]
int qtd[10][10]; // reserva 100 posições para qtd 10 linhas [0 a 9] e 10 colunas [0 a 9]
```

# Arranjo duas Dimensão - Matriz

**Exemplo: -** Programa declarar, zerar e imprimir uma matriz de 5 linha e 3 colunas

Deve-se observar que na manipulação de uma matriz do tipo vetor – uma dimensão , é utilizada uma única instrução de laco (enquanto ou para).

No caso de matrizes com mais dimensões, deve ser utilizado o numero de laços relativo ao tamanho de sua dimensão.

Desta forma, uma matriz de duas dimensões deve ser controlada com dois laços.

**Exemplo: -** Programa declarar, zerar e imprimir uma matriz de 5 linha e 3 colunas

```
var
mat: conjunto [0..4,0..2] de inteiro
 i,j:inteiro
inicio
   para i de 0 ate 4 passo 1 faca
      para j de 0 ate 2 passo 1 faca
          mat[i,j]<-0
      fimpara
   fimpara
   // imprimindo a matriz
   para i de 0 ate 4 passo 1 faca
      para j de 0 ate 2 passo 1 faca
          escreva(mat[i,j]," ")
      fimpara
      escreval
   fimpara
fimalgoritmo
```

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

int main()
{
    int mat[5][3],i,j;
    for (i=0;i<5;i=i+1)
    {
        for (j=0;j<3;j=j+1)
        {
            mat[i][j]=0;
        }
    }
    printf("imprimindo a matriz\n");
    for (i=0;i<5;i=i+1)
    {
        for (j=0;j<3;j=j+1)
        {
            printf("mat[%d][%d]=%d  ",i,j,mat[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

```
imprimindo a matriz
mat[0][0]=0  mat[0][1]=0  mat[0][2]=0
mat[1][0]=0  mat[1][1]=0  mat[1][2]=0
mat[2][0]=0  mat[2][1]=0  mat[2][2]=0
mat[3][0]=0  mat[3][1]=0  mat[3][2]=0
mat[4][0]=0  mat[4][1]=0  mat[4][2]=0
```

# Arranjo duas Dimensão - Matriz

Leia três valores a,b,c. em seguida preencha uma matriz (5x5) da seguinte forma. Os valores acima da diagonal principal deverá ser preenchido com o valor a, abaixo da diagonal principal deverá ser preenchido com o valor b, e a diagonal principal deverá ser preenchida com o valor c.

Leia três valores a,b,c. em seguida preencha uma matriz (5x5) da seguinte forma. Os valores acima da diagonal principal deverá ser preenchido com o va…chido com o valor b, e a diagonal principal deverá ser preenchida

```
var
mat: conjunto [0..4,0..4] de inteiro
 a,b,c,i,j:inteiro
inicio
   leia(a,b,c)
   para i de 0 ate 4 passo 1 faca
      para j de 0 ate 4 passo 1 faca
          se i<j entao
             mat[i,j]=a
          senao
             se i>j entao
               mat[i,j]=b
             senao
               mat[i,j]=c
             fimse
          fimse
      fimpara
   fimpara
   // imprimindo a matriz
   para i de 0 ate 4 passo 1 faca
      para j de 0 ate 4 passo 1 faca
          escreva(mat[i,j]," ")
      fimpara
      escreval
   fimpara
fimalgoritmo
```

```c
int main()
{
    int mat[5][5],a,b,c,i,j;
    printf("Digite o valor acima da diagonal principal:");
    scanf("%d",&a);
    printf("Digite o valor abaixo da diagonal principal:");
    scanf("%d",&b);
    printf("Digite o valor da diagonal principal:");
    scanf("%d",&c);


    for (i=0; i<5; i=i+1)
    { for (j=0; j<5; j=j+1)
        { if (i<j)
            {
                mat[i][j]=a;
            }
            else
            { if (i>j)
                {
                    mat[i][j]=b;
                }
                else
                {
                    mat[i][j]=c;
                }
            }
        }
    }
    printf("imprimindo a matriz\n");
    for (i=0; i<5; i=i+1)
    { for (j=0; j<5; j=j+1)
        {
            printf("%d  ",mat[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```
Digite o valor acima da diagonal principal:1
Digite o valor abaixo da diagonal principal:2
Digite o valor da diagonal principal:3
imprimindo a matriz
3  1  1  1  1
2  3  1  1  1
2  2  3  1  1
2  2  2  3  1
2  2  2  2  3
```

# Alocação dinâmica de memória – Matrizes – Exercício

Desenvolver um programa que leia o número de alunos em uma turma.
Em seguida:
a) criar uma função que devolva um vetor com os nomes dos alunos.
b) uma função que devolva uma matriz contendo quatro notas  por aluno – notas do tipo inteiro.
c) uma função que receba a matriz de notas e devolva um vetor do tipo real contendo a média de cada aluno.
d) uma função que devolva um vetor com a classificação em ordem alfabética dos alunos da turma.
e) uma função que receba os vetores e matrizes criados e imprima os alunos em ordem alfabética , bem como suas médias

# Exercício

Desenvolver um programa que leia o número de alunos em uma turma.
Em seguida:
a) criar uma função que devolva um vetor com os nomes dos alunos.
b) uma função que devolva uma matriz contendo quatro notas  por aluno – notas do tipo inteiro.
c) uma função que receba a matriz de notas e devolva um vetor do tipo real contendo a média de cada aluno.
d) uma função que devolva um vetor com a classificação em ordem alfabética dos alunos da turma.
e) uma função que receba os vetores e matrizes criados e imprima os alunos em ordem alfabética , bem como suas médias

Como não sabemos o número de alunos, vamos trabalhar com alocação dinâmica de memória!

# a) criar uma função que devolva um vetor com os nomes dos alunos.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <string.h>
int main()
{
    setlocale(LC_ALL,"portuguese");
   int nrAlunos;
   char **nomeAlunos;
   printf("Digite o Número de alunos:");
   scanf("%d",&nrAlunos);
   nomeAlunos=lerAlunos(nrAlunos);
```

Teremos um vetor de vetores!

| | | | | | |
|---|---|---|---|---|---|
| J | O | S | E | | |
| M | A | R | I | A | |
| J | O | A | O | | |
| A | N | A | | | |
| L | I | V | I | A | |

```
char ** lerAlunos(int nrAlunos)
{
   char **alunos;
   int i;
   alunos=malloc(sizeof(char *)*nrAlunos);
   // para cada linha será criada um espaço para armazenar os nomes
   // vamos considerar 50 posições para cada nome de aluno
   for (i=0;i<nrAlunos;i=i+1)
   {
      alunos[i]=malloc(sizeof(char)*50);
   }
   for (i=0;i<nrAlunos;i=i+1)
   {
      printf("Digite o nome do %d aluno:",i+1);
      fflush(stdin);
      gets(alunos[i]);
   }
   return alunos;
}
```

# b) uma função que devolva uma matriz contendo quatro notas  por aluno – notas do tipo inteiro.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
#include <string.h>
const int QTDNOTAS=4;
int main()
{
   setlocale(LC_ALL,"portuguese");
  int nrAlunos;
  char **nomeAlunos;
  int **notas;
  printf("Digite o Número de alunos:");
  scanf("%d",&nrAlunos);
  nomeAlunos=lerAlunos(nrAlunos);
  notas=leNotas(nrAlunos,nomeAlunos);
```

Constante QTDNOTAS  para caso o número de notas variar alternar apenas o valor.

Teremos um vetor de vetores!

| N1 | N2 | N3 | N4 |
|---|---|---|---|
| 5 | 6 | 6 | 8 |
| 7 | 6 | 8 | 4 |
| 8 | 8 | 8 | 8 |
| 2 | 8 | 9 | 7 |

```
int ** leNotas(int nrAlunos, char ** alunos)
{
  int **notasAlunos;
  int i,j;
  notasAlunos=malloc(sizeof(int *)*nrAlunos);
  //em linha será criada QTDNOTAS colunas para armazenar as notas
  for(i=0;i<nrAlunos;i=i+1)
  {
    notasAlunos[i]=malloc(sizeof(int)*QTDNOTAS);
  }
  //lendo as notas
  for (i=0;i<nrAlunos;i=i+1)
  {
    printf("Aluno: %s\n ",alunos[i]);
    for (j=0;j<QTDNOTAS;j=j+1)
    {
      printf("digite a %d nota:",j+1);
      scanf("%d",&notasAlunos[i][j]);
    }
  }
  return notasAlunos;
}
```

c) uma função que receba a matriz de notas e devolva um vetor do tipo real contendo a média de cada aluno.

```
int main()
  {
     setlocale(LC_ALL,"portuguese");
    int nrAlunos;
    char **nomeAlunos;
    int **notas;
    float *media;
    printf("Digite o Número de alunos:");
    scanf("%d",&nrAlunos);
    nomeAlunos=lerAlunos(nrAlunos);
    notas=leNotas(nrAlunos,nomeAlunos);
    media=calculaMedia(nrAlunos,notas);
```

Constante QTDNOTAS para caso o número de notas variar alternar apenas o valor.

Teremos um vetor com as médias

| N1 |
|---|
| 5.2 |
| 7.3 |
| 8.0 |
| 4.5 |

```
float * calculaMedia(int nrAlunos,int **notas)
{
   float *mediaNotas;
   int i,j;
   float soma,media;
   mediaNotas=malloc(sizeof(float)*nrAlunos);
   for (i=0;i<nrAlunos;i=i+1)
   {
      soma=0;
      for (j=0;j<QTDNOTAS;j=j+1)
      {
         soma=soma+(float) notas[i][j];
      }
      media=soma/QTDNOTAS;
      mediaNotas[i]=media;
   }
   return mediaNotas;
}
```

d) uma função que devolva um vetor com a classificação em ordem alfabética dos alunos da turma.

```
int main()
  {
     setlocale(LC_ALL,"portuguese");
    int nrAlunos;
    char **nomeAlunos;
    int **notas;
    float *media;
    int *classificacao;
    printf("Digite o Número de alunos:");
    scanf("%d",&nrAlunos);
    nomeAlunos=lerAlunos(nrAlunos);
    notas=leNotas(nrAlunos,nomeAlunos);
    media=calculaMedia(nrAlunos,notas);
classificacao = classificaAlunos(nrAlunos,nomeAlunos);
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | J | O | S | E | | |
| 1 | M | A | R | I | A | |
| 2 | J | O | A | O | | |
| 3 | A | N | A | | | |
| 4 | L | I | V | I | A | |

Teremos um vetor com os índices em ordem alfabética

| índice |
|---|
| 3 |
| 2 |
| 0 |
| 4 |
| 1 |

# d) uma função que devolva um vetor com a classificação em ordem alfabética dos alunos da turma.

Teremos um vetor com os índices em ordem alfabética

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | J | O | S | E | | |
| 1 | M | A | R | I | A | |
| 2 | J | O | A | O | | |
| 3 | A | N | A | | | |
| 4 | L | I | V | I | A | |

| índice |
|---|
| 3 |
| 2 |
| 0 |
| 4 |
| 1 |

int strcmp(const char *str1, const char *str2)

se Return value < 0 então  str1 é menor que str2.
se Return value > 0 então  str1 é maior que  str2.
se Return value = 0 então str1 é igual  a str2.

```
int * classificaAlunos(int nrAlunos,char **nomeAlunos)
{
   int i,j,aux;
   int * nomeClassificado;// irá armazenar apenas o indice do nome.
   // utilizando o método de classificação
   nomeClassificado=malloc(sizeof(int)*nrAlunos);
   for (i=0;i<nrAlunos;i=i+1)
   {
      nomeClassificado[i]=i;
   }

   for (i=0;i<nrAlunos;i=i+1)
   {
      for (j=i+1;j<nrAlunos;j=j+1)
      {
         if (strcmp(nomeAlunos[nomeClassificado[i]],nomeAlunos[nomeClassificado[j]])>0)
         {
            aux=nomeClassificado[i];
            nomeClassificado[i]=nomeClassificado[j];
            nomeClassificado[j]=aux;
         }
      }
   }
   return nomeClassificado;
}
```

e) uma função que receba os vetores e matrizes criados e imprima os alunos em ordem alfabética , bem como suas médias

```
int main()
  {
     setlocale(LC_ALL,"portuguese");
    int nrAlunos;
    char **nomeAlunos;
    int **notas;
    float *media;
    int *classificacao;
    printf("Digite o Número de alunos:");
    scanf("%d",&nrAlunos);
    nomeAlunos=lerAlunos(nrAlunos);
    notas=leNotas(nrAlunos,nomeAlunos);
    media=calculaMedia(nrAlunos,notas);
    classificacao = classificaAlunos(nrAlunos,nomeAlunos);
    imprimeBoletimFinal (nrAlunos, nomeAlunos, classificacao,notas,media);

  return 0;
}
```

e) uma função que receba os vetores e matrizes criados e imprima os alunos em ordem alfabética , bem como suas médias

```
void imprimeBoletimFinal (int nrAlunos, char **alunos, int *classificado, int **notas, float *media)
{
   int i,j;
   printf("Notas Finais em ordem alfabética de nome de aluno\n");
   printf("Ordem \t nome");
   for (i=0;i<QTDNOTAS;i=i+1)
   {
      printf("\t n%d",i+1);
   }
   printf("\t media \n");
   for (i=0;i<nrAlunos;i=i+1)
   {
      printf("%d\t%s",i+1,alunos[classificado[i]]);
      for (j=0;j<QTDNOTAS;j=j+1)
      {
         printf("\t%2d",notas[classificado[i]][j]);
      }
      printf("\t%6.2f\n",media[classificado[i]]);
   }
}
```

PUC Minas
Virtual