# Unidade 5: Vetores e Matrizes

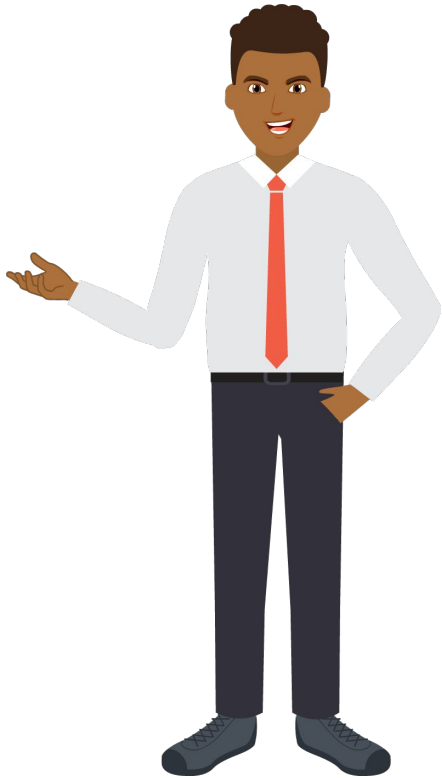
## Introdução

Caro Aluno,

Bem-vindo à disciplina “Vetores e Matrizes”. O objetivo desta disciplina é trabalhar a estrutura de alocação de memória dos vetores e matrizes usando a linguagem C. A partir do estudo desta unidade você será poderá alcançar os seguintes objetivos de aprendizagem:

- Compreender o conceito de vetores e matrizes;
- Criar vetores e matrizes na linguagem C.

Bons estudos!

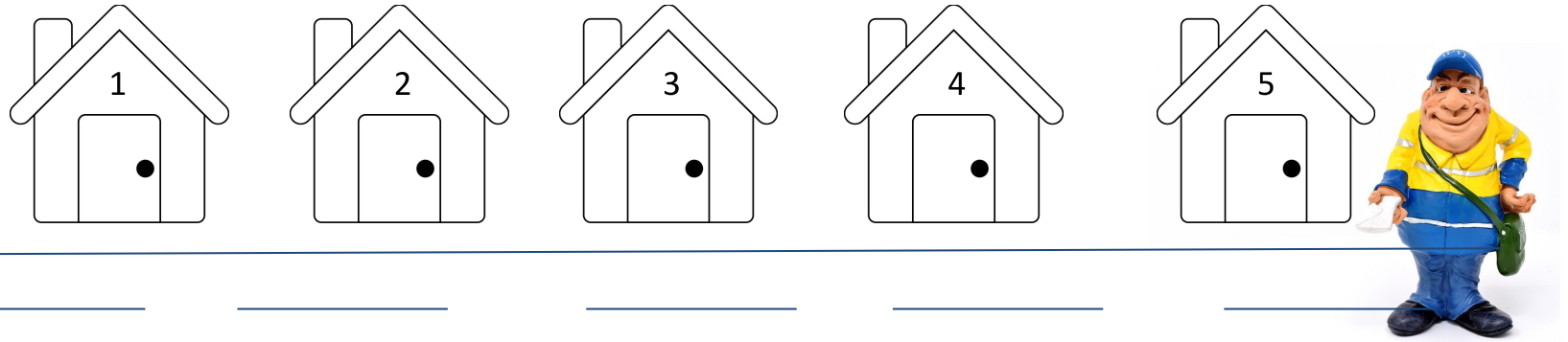


# 4.1 Contextualização de Vetores

Muita gente vê vetores na escola, porém, às vezes, esta ideia é um pouco confusa. Na física, vemos vetores sendo representados como setas para lá e para cá. Na matemática, vemos algo com orientação e sentido. Embora tenhamos conceitos basicamente idênticos, vamos apresentar para você uma definição que você já conhece. Assim fica mais fácil!

O que você vê na figura abaixo?

Um carteiro tentando entregar uma carta?

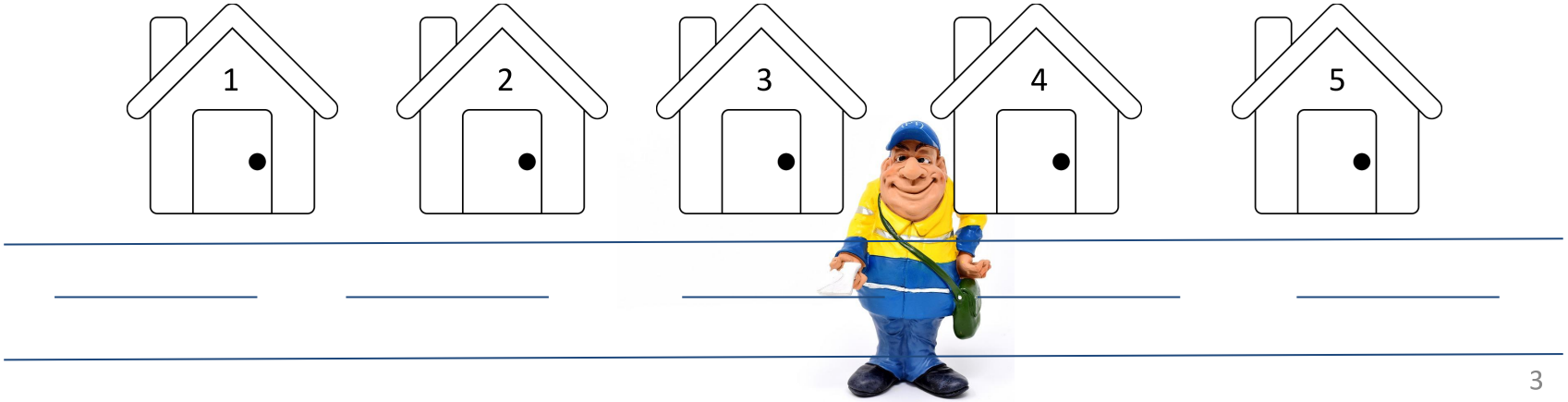


# 4.1 Contextualização de Vetores

Neste cenário, só há uma rua na cidade. O que o carteiro precisa saber para entregar a carta no lugar correto?

Resposta: O número da casa!

Concorda?



# 4.1 Contextualização de Vetores

Em analogia, entenda o vetor como uma única rua em uma cidade. A única coisa que o carteiro precisa saber para chegar ao destino é o número da casa.

Neste contexto, se o carteiro precisar entregar alguma coisa, o que ele precisa saber? O número da casa!

E se ele precisar buscar uma encomenda? O que ele precisa? O número da casa!

Neste cenário, a rua é o meu vetor e a número da casa é a minha chave de identificação de cada casa. Quando formos trabalhar com vetores não será diferente!

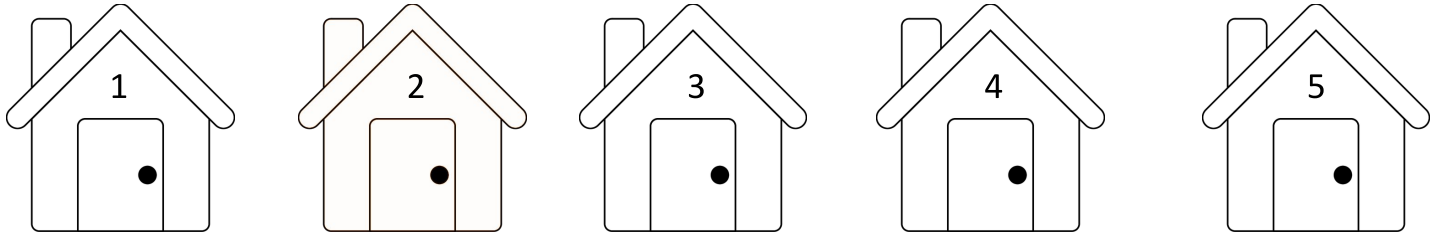
# 4.1 Contextualização de Vetores

Em uma analogia mais simples, entenda o vetor como uma estrutura com endereços e em cada endereço uma “casa” que pode receber o tipo de variável que você definir.

Sendo assim, ao percorrer o “vetor” você pode armazenar, consultar, excluir e modificar o conteúdo dentro de cada “casinha”.

Viu como você sabe o conceito de vetor?

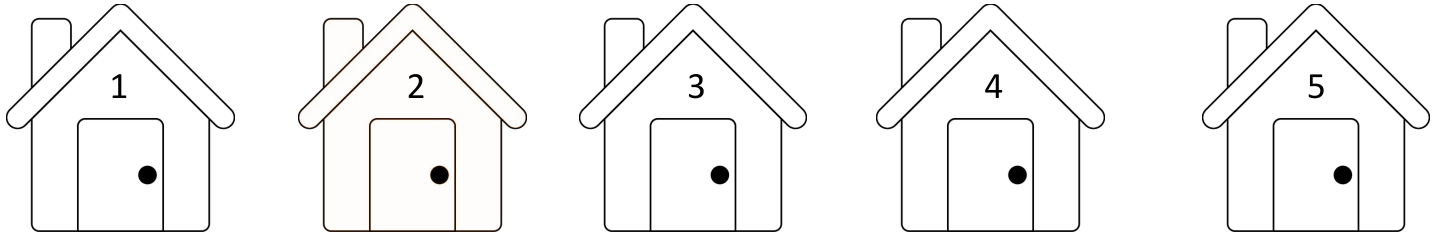
Agora vamos partir para parte técnica!



# 4.1 Contextualização de Vetores

Atenção: na língua inglesa, o nome “vetores” é “array”.

Não se assuste se alguém utilizar o termo array para se referir a vetores. É a mesma coisa, mas em línguas distintas.  Você vai se deparar com isso frequentemente.



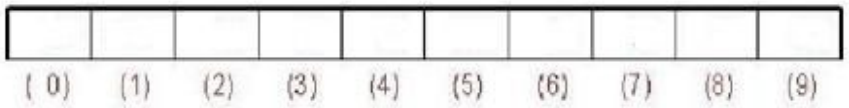
# 4.2 Vetores

O vetor é uma estrutura de dados indexada, que pode armazenar uma determinada quantidade de valores do mesmo tipo. Os dados armazenados em um vetor são chamados de itens do vetor. Para localizar a posição de um item em um vetor usamos um número inteiro denominado índice do vetor.

## 4.2.1 Vantagem de utilização do vetor

Facilidade de manipular um grande conjunto de dados do mesmo tipo declarando-se apenas uma variável.

## 4.2.2 Representação gráfica de um vetor.



## 4.2.3 Vetor em linguagem C

Sintaxe:

Tipo NomeDoVetor[quantidade_de_itens];

Exemplo:

Declaração do vetor do tipo float com 10 números:

float V[10];

É importante notar que em linguagem C, o vetor é indexado a partir da posição zero.

Podemos dizer que em C :

- A primeira posição de um vetor tem índice zero.
- A última posição de um vetor tem índice = número de posições – 1.

## 4.2.4 Declarando e inicializando de vetores

**Exemplo 1:**

Podemos declarar e inicializar um vetor com um tamanho constante, como abaixo:

int numeros[5] = {10, 20, 30, 40, 50};

**Exemplo 2:**

Iniciando apenas alguns elementos do vetor:

int valores[5] = {2,4,6};

será equivalente a

int valores[5] = {2,4,6,0,0};

Isto ocorre porque apenas alguns itens do vetor foram inicializados.

Neste caso, quando o número de itens inicializados é menor que o número total de itens do vetor, os itens não inicializados são automaticamente zerados.

**Exemplo 3:**

Inicializando um vetor sem especificar a quantidade de elementos

```
int valores[] = {3,5,7};
```

Neste exemplo, não foi especificado o tamanho do vetor, porém ao inicializar os elementos o compilador faz a contagem dos itens e determina o tamanho do vetor automaticamente.

# Programa usando vetor em C

```
#include<stdio.h>
#include<conio.h>
  int main(void)
  {
    float notas[5] = {7, 8, 9.5, 9.9, 5.2};
    // declarando e inicializando o vetor notas

    printf("Exibindo os Valores do Vetor \n\n");
    printf("notas[0] = %.1f\n", notas[0]);
    printf("notas[1] = %.1f\n", notas[1]);
    printf("notas[2] = %.1f\n", notas[2]);
    printf("notas[3] = %.1f\n", notas[3]);
    printf("notas[4] = %.1f\n", notas[4]);

     return 0;
}
```

**Explicação do código**

Para fazer referência a uma determinada posição do vetor, devemos utilizar o nome do array e seu respectivo índice.

Por exemplo:

notas[0], faz referência ao elemento armazenado no vetor notas posição (índice) zero.

Para exibir esse elemento na tela usamos:

printf(“notas[0] = %.1f\n”, notas[0]);

**Utilizando um laço for para automatizar a exibição de um vetor**

Programa usando vetor em C

```
#include<stdio.h>
#include<conio.h>
int main(void)
{

   int i;
   float notas[5] = {7, 8, 9.5, 9.9, 5.2};
   // declarando e inicializando o vetor notas

   printf("Exibindo os Valores do Vetor \n\n");

   for( i = 0 ; i <= 4; i++)
   {
      printf("notas[%d] = %.1f\n",i, notas[i]);
   }

   getch();
   return 0;
 }
```

**Explicação do código**

Declaramos uma variável de controle para o laço for.

Observe que a variável i, tem uma dupla função neste programa.

Além de controlar o laço a variável i também foi usada como índice do vetor.

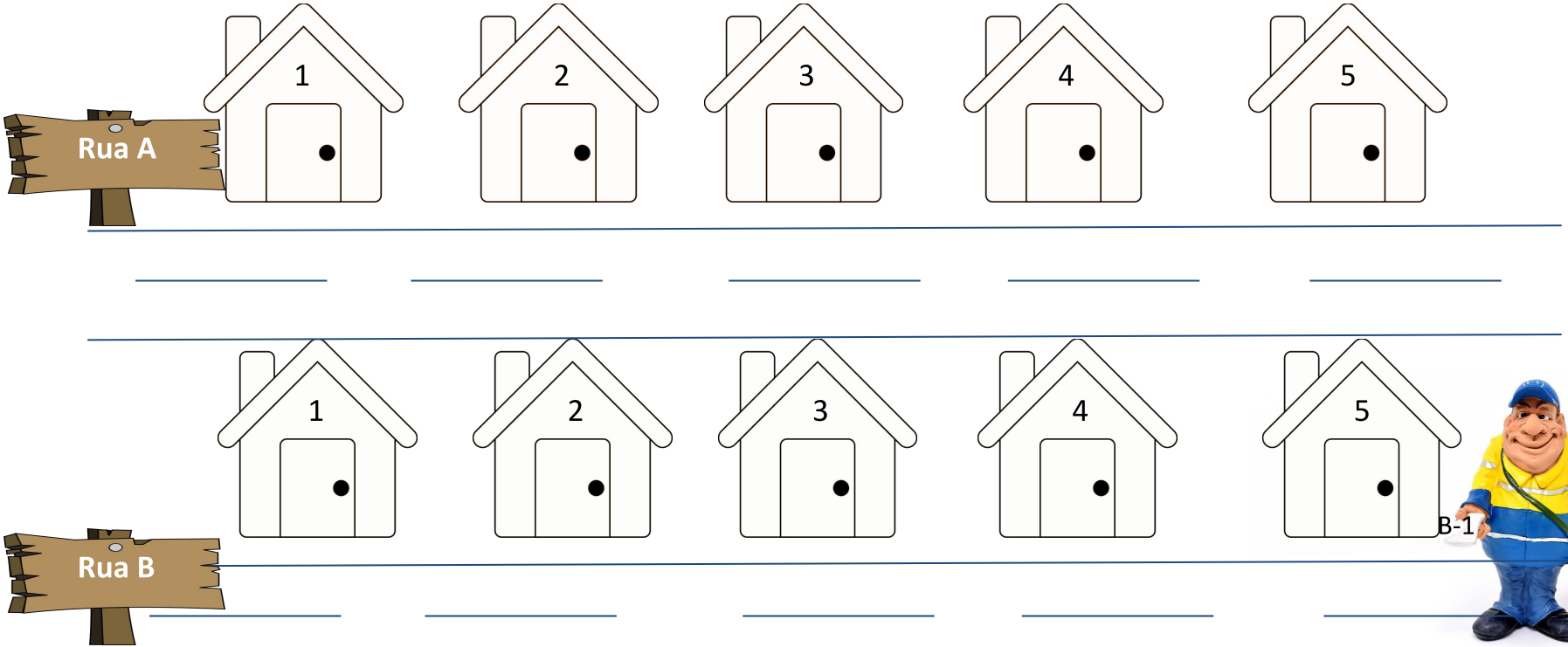
Desta forma, podemos inicializar i com valor zero e fazê-la variar até o tamanho do vetor – 1, ou seja, a variável parte de zero e vai até 4, exatamente como os índices do vetor.

Como i varia de zero até 4, faz um total de 5 repetições no controle do laço, e ao mesmo tempo, controla a posição do vetor cujo elemento será mostrado na tela.

# 4.3 Contextualização de Matrizes

O que você vê na figura abaixo?

Se um carteiro precisa entregar uma carta agora, o que ele precisa? Apenas o número da casa?



# 4.3 Contextualização de Matrizes

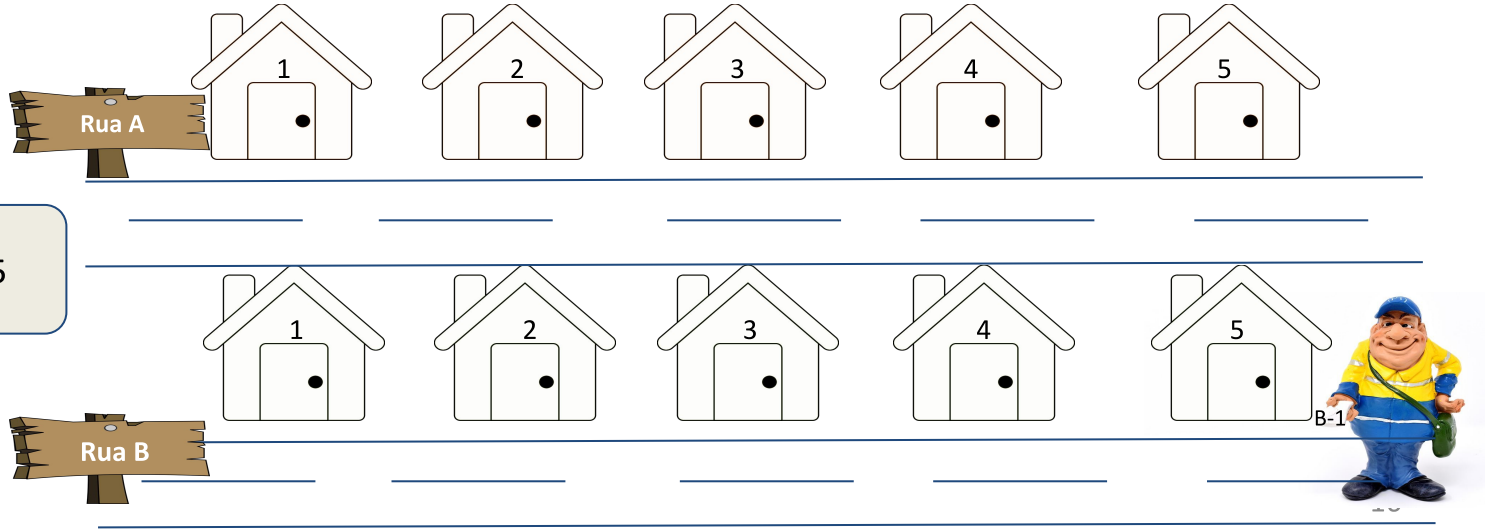
Veja que neste cenário, há **duas** dimensões.

Agora, para o carteiro poder chegar ao destino correto, ele precisa de dois elementos, uma “chave” para a rua e uma “chave” para a casa.

Agora, a única maneira de possível de chegar até o destino correto é através de **dois índices**!

**Entendemos por matrizes os vetores com mais de uma dimensão!**

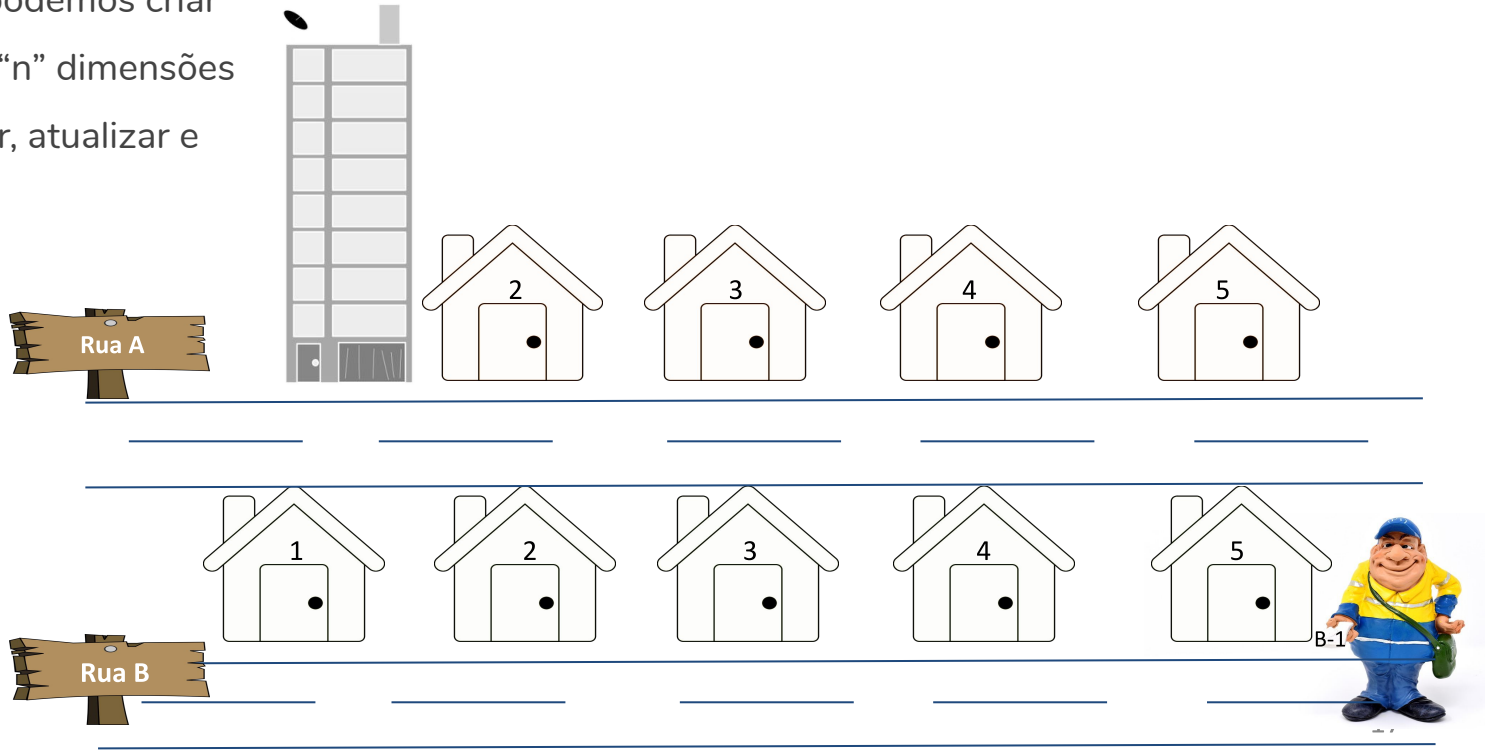
Nossa matriz aqui é 2:5



# 4.3 Contextualização de Matrizes

No cenário abaixo, quantas dimensões e quantos índices eu preciso neste sistema para se chegar até um destino?

Perceba que podemos criar matrizes com “n” dimensões para inserir, ler, atualizar e deletar dados.



# 4.3 Contextualização de Matrizes

E nesta matriz? Quantas dimensões eu tenho?

| | Nome | Idade | Informações Técnicas | |
|---|---|---|---|---|
| 1 | Ana | 18 | Nível | C |
| | | | País | França |
| | | | Escolaridade | Médio |
| | | | Formação | Arquitetura |
| 2 | João | 32 | Nível | B |
| | | | País | Brasil |
| | | | Escolaridade | Superior |
| | | | Formação | Artes |

# 4.4 MATRIZES

Uma Matriz é um conjunto de variáveis de mesmo tipo que compartilham um mesmo nome. Com Matriz agora podemos armazenar valores para depois serem manipulados através de índices, o qual referência cada um dos elementos. Para se criar uma Matriz é necessário definir um tipo, seu nome e quantidade de elementos, sendo este último entre colchetes ([ ]). O número de dimensões depende da quantidade de colchetes.

Vejamos agora a declaração de uma matriz.

Exemplo: int mat[5][12];  //criamos uma matriz bidimensional

int matmuli[5][12][3];  //criamos uma matriz tridimensional

## 4.4.1 Referenciação e Atribuição dos elementos de uma matriz

Para se referenciar um elemento da Matriz individualmente, basta apenas colocar-se o nome desta Matriz seguida do seu índice entre colchetes. Os índices dos elementos de uma matriz são sempre iniciados por zero.

## 4.4.1 Sintaxe

A sintaxe para criar a matriz é informando o tipo da variável e entre colchetes o tamanho de cada vetor dentro da matriz.

Exemplo:

```
tipo nome_matriz[tamanho_da_dimensão_2_vetor][tamanho_da_dimensão_1_vetor]
```

Declaração real:

```
int matriz[10][20]  //é criado uma matriz bidimensional que a segunda dimensão tem 10 elementos e que a primeira dimensão tem 20 elementos
```

Observação importante: a matriz declarada com a 2ª dimensão de 10 elementos e a 1ª dimensão com 20 elementos começarão com valores a partir de “0”. Veremos a frente

## 4.4.2 Instanciando matrizes

Vejamos um exemplo para se referenciar e atribuir valores em uma elemento da Matriz:

```
int mat[10][5];

mat[0][0] = 20; /* o elemento de mat na posição na dimensão 1:1 recebe o valor 20 */
mat[1][2] = 5; /* o elemento de mat na posição na dimensão 2:3 recebe o valor 5 */
....
mat[2][4] = 7; /* o elemento de mat na posição na dimensão 3:5 recebe o valor 7 */
...
mat[9][4] = 12; /* o elemento de mat na última posição da matriz, ou seja, na dimensão 10:5
                   recebe o valor 12 */
x = mat[9][4]; /* x recebe o elemento de matriz da posição na dimensão 10:5  que vale 12 */
```

## 4.4.3 Inicialização de matrizes

Para se inicializar uma Matriz, ou seja, atribuir valores dentro da matriz na sua declaração, é necessário apenas colocar-se o nome desta Matriz, a quantidade elementos entre colchetes seguida do operador de atribuição e por fim os valores separados por vírgula entre as chaves ({ }).

```
int matrx [3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };  /* matriz inicializada com os valores de 1 a 12 */
```

Para exemplificar melhor, vejamos o exemplo abaixo de um programa que calcula a média de três notas em uma matriz bi-dimensional sendo que um dos índices refere-se a disciplinas distintas.
Exemplo:
Disciplina 1 - Matemática
Disciplina 2 - Física

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

int main() {
int i,j;

float nota[2][3], m=0;

        for (i=0;i<2;i++) {
                for (j=0;j<3;j++) {
                printf("\n Digite a nota %d da disciplina %d ", j+1, i+1  );
                scanf("%f",&nota[i][j]);
                m = m + nota[i][j];
                }

        printf("\n A media da disciplina %d e' %.2f \n", i+1, m/3 );
        m = 0;  //depois de imprimir o valor em tela zera a variável para o cálculo da média da próxima disciplina
        }
 system("pause");
 return 0;
}
```

# Resumo

Nesta unidade continuamos estudando sobre a Linguagem C, porém buscando compreender conceitual e funcionalmente os vetores e as matrizes por meio da construção e interpretação dos mesmo em suas diferentes representações.

# Hora de Revisar!

Parabéns! Você terminou o estudo desta Unidade.

Que tal fazermos uma retomada resumida dos conteúdos estudados apenas para te ajudar a sintetizar o que você estudou? Então, vamos lá!

- Conceitos de vetores e matrizes
- Criar programas simples usando Vetores e/ou matrizes