



# LÓGICA DE PROGRAMAÇÃO E ALGORITMOS

AULA 5



Prof. Sandro de Araujo



## CONVERSA INICIAL

Esta aula teve como base os livros: Lógica de programação e estruturas de dados, Fundamentos da Programação de Computadores e Treinamento em Linguagem C. Em caso de dúvidas ou aprofundamento, consulte-os em nossa Biblioteca Virtual Pearson.

Esta aula apresenta a seguinte estrutura de conteúdo:

1. Vetor e matriz
2. Declarar e inicializar vetores
3. Declarar e inicializar matrizes
4. Cadeia de caracteres
5. Inicializar uma cadeia de caracteres

O objetivo desta aula é conhecer os principais conceitos e aplicações de vetor, matriz e cadeia de caracteres, como declará-los e inicializá-los. Bem como representá-los em pseudocódigo e linguagem C para resolver problemas computacionais.

## TEMA 1 – VETOR E MATRIZ

Vetor ou array é uma variável composta homogênea unidimensional. Isso quer dizer que se trata de um conjunto de variáveis do mesmo tipo, que possuem o mesmo identificador (nome) e são alocadas sequencialmente na memória (Ascencio, 2012). Como as variáveis têm o mesmo nome, para localizar a posição de um item em um vetor, usamos um número inteiro, chamado de índice, para referência de sua localização dentro da estrutura.

Exemplo de declaração de um vetor usando linguagem de programação C:

- `int temp[10];`

A Figura 1, a seguir, mostra a estrutura representativa do exemplo anterior, um vetor de números inteiros.



Figura 1 – Representação de um vetor

índice:	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
nome: temp	18	17	20	26	32	29	15	12	16	21

Fonte: Puga e Rissetti, 2016.

No exemplo da Figura 1 temos um vetor com o nome de **temp**, do tipo **inteiro**, com **dez** posições. Onde, **18** é o elemento com índice **0**, **17** é o elemento com índice **1**, **20** é o elemento com índice **2** e assim por diante.

Para manipular um determinado elemento em um vetor, é necessário fornecer o identificador (nome) e o índice do elemento, é o índice que determina a posição na qual o elemento se encontra. Cada elemento pode ser manipulado individualmente, basta indicar sua posição usando o índice (Puga; Rissetti, 2016).

Já uma Matriz é um vetor de vetores. Assim como no vetor, cada item da matriz é acessado por um número inteiro chamado de índice, que armazena vetores em uma estrutura de dados com duas ou mais dimensões (Ascencio, 2012).

Em uma matriz, todos os itens também têm que ser do mesmo tipo de dado para formar suas tabelas na memória.

Exemplo de declaração de matriz com 2 dimensões usando linguagem C

- `int temp[3][2];`

A Tabela 1 mostra a estrutura representativa do exemplo acima, um vetor de números inteiros.

Tabela 1 – Representação de uma matriz

ÍNDICES	DADOS
[0,0]	19
[0,1]	21
[1,0]	33
[1,1]	581



Nesse exemplo, o valor 3 representa a quantidade de linhas e o valor 2 representa a quantidade de colunas, compondo uma matriz do tipo 3 X 2 com capacidade de armazenamento de até seis elementos (itens) do tipo **int** (inteiro). Para cada dimensão da matriz é necessário utilizar um índice para posicionar a linha, outro para a coluna. Daí o nome **matriz bidimensional** (dois índices).

## TEMA 2 – DECLARAR E INICIALIZAR VETORES

Declarar um vetor é dar um nome seguindo uma sintaxe pré-estabelecida pela linguagem de programação que será usada para a construção do algoritmo. A declaração de um vetor deve conter três informações:

- 1) Nome do vetor (identificador)
- 2) Número de posições (tamanho)
- 3) Tipo de dado que será armazenado

Quando declaramos um vetor de tamanho quatro, alocamos na memória do computador quatro posições para armazenar os elementos com o mesmo nome. E o termo inicializar um vetor é colocar elementos nos espaços nesses alocados na memória.

Para inicializar os valores 7.6, 8.9, 9 e 9.9 em um vetor com o nome *nota*, teremos a seguinte estrutura, mostrada na Tabela 2.

Tabela 2 – Representação da estrutura de um vetor com quatro posições

Índice:	[0]	[1]	[2]	[3]
Nome: nota	7.6	8.9	9	9.9

O exemplo a seguir mostra a sintaxe do vetor anterior na representação algorítmica em pseudocódigo:

- NomeDoVetor: Vetor[início..final] de TipoDoVetor

O “**NomeDoVetor**” é o identificador, “**Vetor[início..final]**” define o início e o fim da posição dos elementos e “**de TipoDoVetor**” é o tipo da coleção de variáveis.

Exemplo:

- Declaração do vetor do tipo inteiro com 4 posições em pseudocódigo:



- nota: vetor[1..4] de Inteiro

Esta declaração define um vetor chamado “**nota**”, que armazena um conjunto de números inteiros identificados como nota[1], nota[2], nota[3] e nota[4]. Temos, então, um conjunto de números inteiros, cada qual em um endereço sequencial diferente, identificado pelo índice do vetor e localizado dentro dos colchetes [ ]. Desta forma, nota[1] guarda o primeiro número inteiro, nota[2] guarda o segundo número inteiro e assim sucessivamente, até chegar no nota[4], que contém o último número inteiro que será armazenado.

A sintaxe na linguagem C segue a mesma lógica com uma única diferença, o índice começa com o número “0” zero. A seguir, veremos a sintaxe da declaração de um vetor em C:

- TipoDoVetor NomeDoVetor[QuantidadeDeElementos];

Exemplo:

- Declaração do vetor do tipo inteiro com 4 posições na linguagem de programação C:
- int nota[4];

Esta declaração define um vetor chamado “**nota**”, que armazena um conjunto de números do tipo “**int**” (inteiros) identificados como nota[0], nota[1], nota[2] e nota[3]. Desta forma, nota[0] que vai guardar o primeiro número inteiro, nota[1] guarda o segundo número inteiro e assim sucessivamente, até chegar no nota[3], que contém o último número inteiro que será armazenado.

Para esse exemplo, temos os seguintes índices: 0, 1, 2 e 3, totalizando quatro posições.

É importante ressaltar que na linguagem de programação C, o vetor é indexado a partir da posição zero.

Veja o exemplo a seguir:

- int números[6] = {1, 2, 3, 4, 5, 6};

O “**int números[6]**” declara um vetor com 6 posições que vai do índice 0 até o índice 5, e o “**= {1, 2, 3, 4, 5, 6};**” com índices 0, 1, 2, 3, 4 e 5, respectivamente.

Também podemos inicializar o vetor com apenas alguns elementos, veja o exemplo a seguir:

- int numeros[6] = {1,2,3};



- o vetor anterior é equivalente a **int numeros[6] = {1,2,3,0,0,0}**.

Quando o número de itens inicializados é menor que o número total de itens do vetor, os itens não inicializados são automaticamente preenchidos com o valor zero.

Também é possível inicializar um vetor sem especificar a quantidade de elementos, Veja o exemplo abaixo:

- `int numerosInteiros[ ] = {3,6,7};`

Ou seja:

- `numerosInteiros[0] = 3;`
- `numerosInteiros[1] = 6;`
- `numerosInteiros[2] = 7;`

Quando não especifica a quantidade de elementos, ao inicializar os elementos, o compilador faz a contagem dos itens e determina o tamanho do vetor automaticamente. Vetor com três elementos terá um tamanho de três posições.

A seguir, dois exemplos de vetor em pseudocódigo e Linguagem C:

### **Exemplo 1:**

Considere um algoritmo que vai imprimir na tela quatro notas armazenadas em um vetor do tipo real.

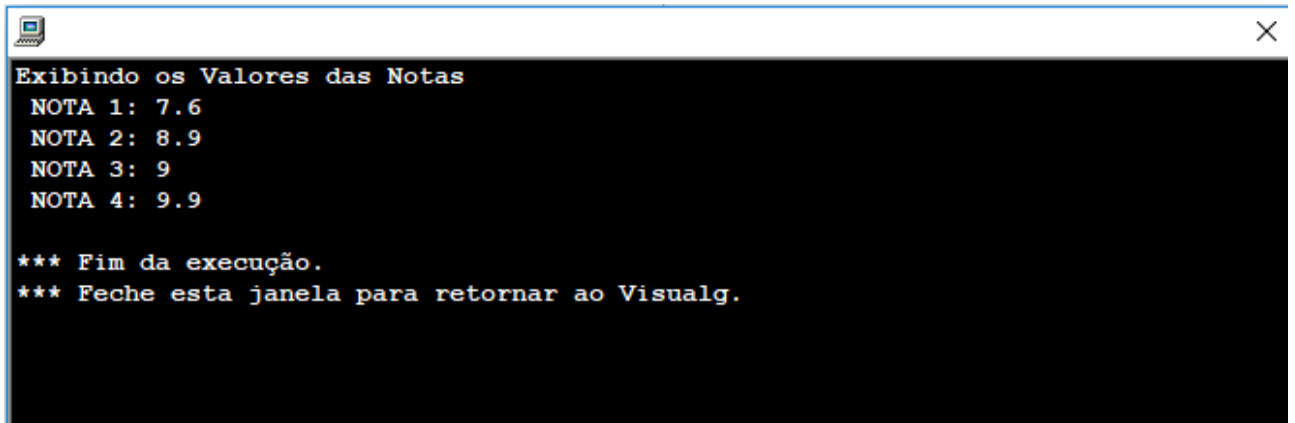
#### **1) Pseudocódigo**

```
algoritmo "exemploVetor"
var
nota: vetor[1..4] de real // declarando o vetor nota
inicio
nota[1] <- 7.6 // inicializando o vetor nota
nota[2] <- 8.9 // inicializando o vetor nota
nota[3] <- 9 // inicializando o vetor nota
nota[4] <- 9.9 // inicializando o vetor nota
escreval ("Exibindo os Valores das Notas")
escreval (" NOTA 1:", nota[1])
escreval (" NOTA 2:", nota[2])
escreval (" NOTA 3:", nota[3])
escreval (" NOTA 4:", nota[4])
```



A Figura 2 mostra a saída do exemplo anterior:

Figura 2 – Saída do algoritmo exemploVetor



```
Exibindo os Valores das Notas
NOTA 1: 7.6
NOTA 2: 8.9
NOTA 3: 9
NOTA 4: 9.9

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

## 2) Linguagem C

```
#include<stdio.h>
#include<conio.h>
int main( )
{
    float nota[4] = {7.6, 8.9, 9, 9.9}; // declarando e inicializando o vetor nota
    printf("Exibindo os Valores das Notas \n\n");
    printf(" NOTA 1: %.1f\n", nota[0]);
    printf(" NOTA 2: %.1f\n", nota[1]);
    printf(" NOTA 3: %.1f\n", nota[2]);
    printf(" NOTA 4: %.1f\n", nota[3]);
    system("pause");
    return 0;
}
```

A figura 3 mostra a saída do exemplo anterior:



Figura 3 – Saída do algoritmo na linguagem C

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\teste1\bin\Debug\teste1.exe"
Exibindo os Valores das Notas

NOTA 1: 7.6
NOTA 2: 8.9
NOTA 3: 9.0
NOTA 4: 9.9
Pressione qualquer tecla para continuar. . .
```

### Exemplo 2:

Considere um algoritmo que vai pegar três notas e armazená-las em um vetor. Após verificação do teste, vai imprimir na tela se o aluno foi aprovado ou reprovado e qual foi a sua média.

#### 1) Pseudocódigo

```
algoritmo "Exemplo2"
var
    nota: vetor[1..4] de real
inicio
    escreval("***** Digite a sua PRIMEIRA nota: *****")
    leia(nota[1])
    enquanto (nota[1] < 0) ou (nota[1] > 10) faça
        escreval("ATENÇÃO! digite um número entre 0 e 10 para a PRIMEIRA
        nota.")
        leia(nota[1])
    fimenquanto
    escreval("***** Digite a sua SEGUNDA nota: *****")
    leia(nota[2])
    enquanto (nota[2] < 0) ou (nota[2] > 10) faça
        escreval("ATENÇÃO! digite um número entre 0 e 10 para a SEGUNDA
        nota.")
        leia(nota[2])
    fimenquanto
    escreval("***** Digite a sua TERCEIRA nota: *****")
    leia(nota[3])
```

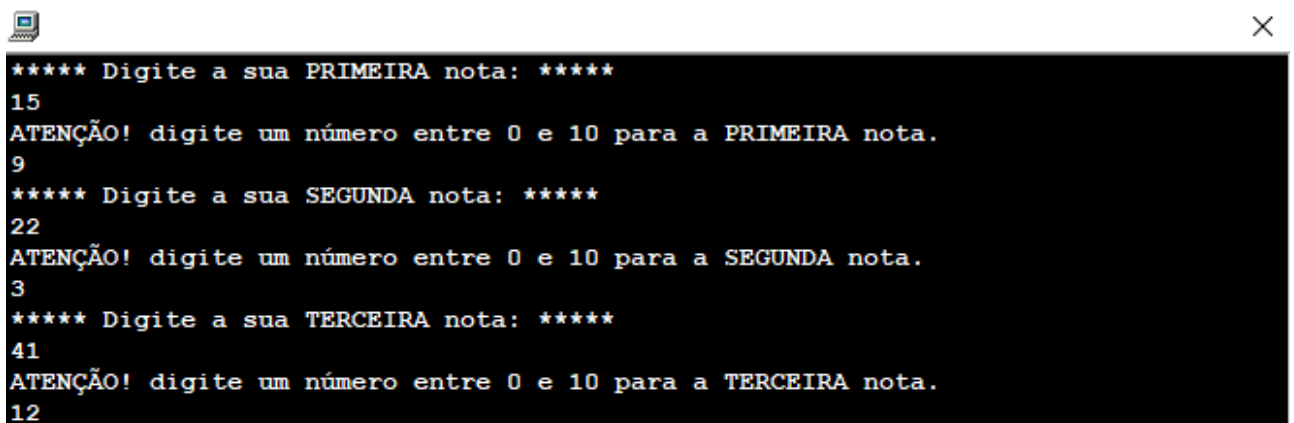




```
enquanto (nota[3] < 0) ou (nota[3] > 10) faca
  escreval("ATENÇÃO! digite um número entre 0 e 10 para a TERCEIRA
  nota.")
  leia(nota[3])
fimenquanto
nota[4] <- ((nota[1] + nota[2] + nota[3])/3)
se (nota[4] >= 7) entao
  escreval("----- PARABÉNS! VOCÊ FOI APROVADO com média: ",
  nota[4], " -----")
senao
  escreval("----- ESTUDE MAIS E NÃO DESISTA! VOCÊ FOI
  REPROVADO com média: ", nota[4], " -----")
fimse
fimalgoritmo
```

Para o exemplo anterior, temos várias saídas que depende de alguns testes que serão realizados. As primeiras saídas acontecem caso o usuário digite um número que não esteja entre o intervalo de 0 a 10. Para as três entradas (nota[0], nota[1] e nota[2]), o algoritmo fará um teste e caso o número digitado esteja fora do intervalo apresentará como saída uma mensagem pedindo para que o usuário digite um número novamente. A Figura 4 mostra as primeiras saídas do exemplo anterior:

Figura 4 – Saída do algoritmo em pseudocódigo



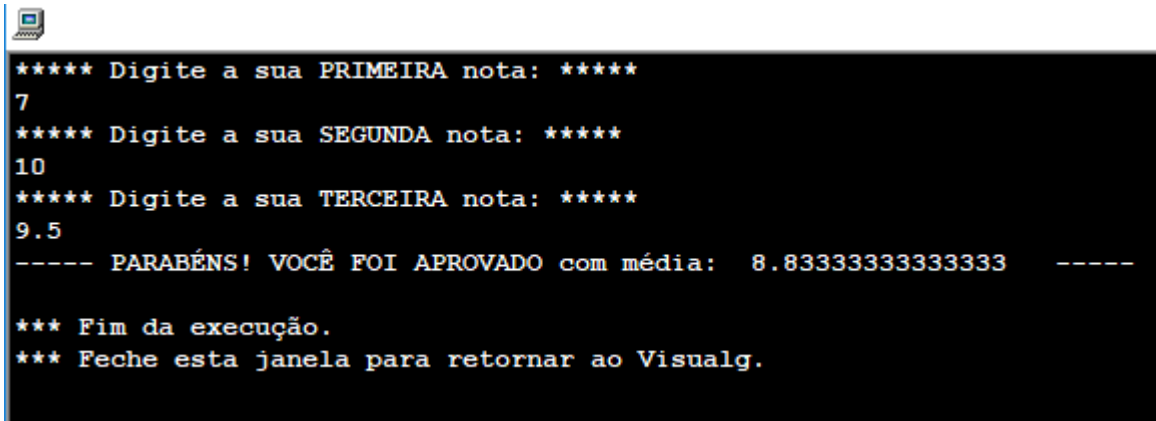
```
***** Digite a sua PRIMEIRA nota: *****
15
ATENÇÃO! digite um número entre 0 e 10 para a PRIMEIRA nota.
9
***** Digite a sua SEGUNDA nota: *****
22
ATENÇÃO! digite um número entre 0 e 10 para a SEGUNDA nota.
3
***** Digite a sua TERCEIRA nota: *****
41
ATENÇÃO! digite um número entre 0 e 10 para a TERCEIRA nota.
12
```

Repare que, enquanto o usuário não digitar um número entre o intervalo 0 e 10, o algoritmo vai continuar repetindo a mensagem para o usuário digitar o número entre o intervalo 0 e 10. Após testar todas as entradas, o algoritmo



calcula a média das notas e executa um novo teste para verificar se o aluno foi aprovado ou reprovado. A Figura 5 mostra a saída desse teste:

Figura 5 – Saída do teste aprovado ou reprovado



```
***** Digite a sua PRIMEIRA nota: *****
7
***** Digite a sua SEGUNDA nota: *****
10
***** Digite a sua TERCEIRA nota: *****
9.5
----- PARABÉNS! VOCÊ FOI APROVADO com média: 8.83333333333333 -----
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Para esse exemplo, usamos apenas uma variável, um vetor, com o nome `nota[ ]`, tanto para as entradas como para as saídas.

Como seria a implementação desse pseudocódigo na linguagem C?

É o que veremos no exemplo a seguir.

## 2) Linguagem C

```
#include <stdio.h>

int main(){
    float nota[4];
    printf("***** Digite a sua PRIMEIRA nota: *****\n");
    scanf("%f", &nota[0]);
    while((nota[0] < 0)|| (nota[0]>10)){
        printf("ATENCAO! digite um numero entre 0 e 10 para a PRIMEIRA
        nota.\n");
        scanf("%f", &nota[0]);
    }
    printf("***** Digite a sua SEGUNDA nota: *****\n");
    scanf("%f", &nota[1]);

    while((nota[1] < 0)|| (nota[1]>10)){

        printf("ATENCAO! digite um numero entre 0 e 10 para a SEGUNDA
        nota.\n");
        scanf("%f", &nota[1]);
    }
    printf("***** Digite a sua TERCEIRA nota: *****\n");
```



```
scanf("%f", &nota[2]);
while((nota[2] < 0)||((nota[2]>10)){
printf("ATENCAO! digite um notaero entre 0 e 10 para a TERCEIRA
nota.\n");
scanf("%f", &nota[2]);
}
nota[3]= (nota[0]+nota[1]+nota[2])/3;
if(nota[3]>=7)
printf("----- PARABENS! VOCE FOI APROVADO com media: %.2f\n\n",
nota[3]);
else
printf("----- ESTUDE MAIS E NAO DESISTA! VOCE FOI REPROVADO
com media: %.2f\n\n", nota[3]);
system("pause");
return 0;
}
```

Pegamos o exemplo usado no pseudocódigo (algoritmo "Exemplo2") e usamos a mesma lógica para resolver o problema com a linguagem de programação C. As primeiras saídas do teste verificaram se o número digitado está fora do intervalo de 0 a 10 e, caso a condição seja verdadeira, aparecerá uma mensagem de erro para que o usuário digite outro número.

Após os testes das entradas de dados, realiza-se o cálculo da média das notas e o resultado vai passar por um novo teste, cujo objetivo é verificar se a média é maior ou igual a 7. Se a média for maior ou igual a 7, o algoritmo vai imprimir na tela do usuário uma mensagem de aprovado acompanhado do valor da média, senão vai imprimir reprovado também acompanhado do valor da média. A Figura 6 mostra as primeiras saídas do algoritmo. Após os primeiros testes, o algoritmo mostrará na tela se o aluno foi aprovado ou reprovado. Conforme mostrado nas Figuras 7 e 8, respectivamente.

Figura 6 – Primeiras saídas do algoritmo na linguagem C

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\vetor\bin\Debug\vetor.exe"
***** Digite a sua PRIMEIRA nota: *****
11
ATENCAO! digite um numero entre 0 e 10 para a PRIMEIRA nota.
10
***** Digite a sua SEGUNDA nota: *****
55
ATENCAO! digite um numero entre 0 e 10 para a SEGUNDA nota.
10
***** Digite a sua TERCEIRA nota: *****
5689
ATENCAO! digite um n-mero entre 0 e 10 para a TERCEIRA nota.
10
----- PARABENS! VOCE FOI APROVADO com media: 10.00,
Pressione qualquer tecla para continuar. . . █
```

Figura 7 – Saída aprovado

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\vetor\bin\Debug\vetor.exe"
***** Digite a sua PRIMEIRA nota: *****
10
***** Digite a sua SEGUNDA nota: *****
10
***** Digite a sua TERCEIRA nota: *****
10
----- PARABENS! VOCE FOI APROVADO com media: 10.00
Pressione qualquer tecla para continuar. . . █
```



Figura 8 – Saída reprovado

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\vetor\bin\Debug\vetor.exe"
***** Digite a sua PRIMEIRA nota: *****
4.5
***** Digite a sua SEGUNDA nota: *****
3.5
***** Digite a sua TERCEIRA nota: *****
7.8
----- ESTUDE MAIS E NAO DESISTA! VOCE FOI REPROVADO com media: 5.27
Pressione qualquer tecla para continuar. . . .
```

Para esse exemplo, também usamos apenas uma variável, um vetor, com o nome `nota[ ]`, tanto para as entradas como para as saídas.

### TEMA 3 – DECLARAR E INICIALIZAR MATRIZES

Para uma matriz, é necessário combinar os dados internos com duas variáveis. Caso precise incluir a matrícula do aluno em um vetor, teremos algo como mostrado na Tabela 3.

Tabela 3 – Representação da estrutura de um vetor com quatro posições

Matrícula	Nota 1	Nota 2	Nota 3	Nota 4
14505	7.6	8.9	9	9.9
45215	7.5	9	8.5	10
89541	9	8.5	9.5	9.5
96541	8.5	7.5	7.5	6

Na Tabela 3 vemos a relação de duas variáveis em qualquer espaço da matriz; se precisar saber a Nota 3 da Matrícula 89541, terá que localizar a linha 89541 da Matrícula e cruzá-la com a coluna Nota 3, onde será localizado o valor 8.5, que corresponde à segunda nota da Matrícula 89541.

O exemplo a seguir mostra a sintaxe da matriz na representação algorítmica em pseudocódigo:

- NomeDaMatriz: Vetor[inicio1..final1, inicio2..final2] de TipoDeVariavel



**NomeDaMatriz** é o identificador, **Vetor[inicio1..final1, ... ]** define o início e o fim da posição dos elementos na linha, **Vetor[ ... , inicio2..final2]** define o início e o fim da posição dos elementos na coluna e **TipoDeMatriz** é o tipo da coleção de variáveis.

Exemplo:

- notas: vetor[1..2, 1..2] de Real;

Na linguagem de programação C, assim como no vetor, a matriz também é indexada a partir da posição zero e segue a mesma lógica do pseudocódigo. A seguir, veremos a sintaxe da declaração de uma matriz em C:

- TipoDaMatriz NomeDaMatriz[QtdeDeLinhas] [QtdeDeColunas];

Exemplo:

- float notas[2][2];

Note que temos duas linhas: notas[0][] e notas[1][], e duas colunas: notas[][0] e notas[][2], em cada linha dessa temos 2 elementos que compõem uma coluna, compondo uma matriz com 4 posições.

Ou seja, é uma matriz de duas linhas e duas colunas.

Sempre o primeiro número é a linha e o segundo número é a coluna.

Para declarar uma matriz 2X4 e inicializá-la, devemos colocar cada linha entre chaves {}, e separá-las por vírgulas:

Exemplo:

- int numeros[2][4] = { {11, 22, 35, 54 }, {5, 9, 16, 8 } };

O “**int numeros[2][4]**” declara uma matriz do tipo inteiro, com o nome **numeros**, e com duas linhas e quatro colunas. A inicialização da matriz com os elementos anteriores ficaria:

11	22	35	54
5	9	16	8

} Duas linhas

Quatro colunas

A seguir, veremos um exemplo de matriz em pseudocódigo e Linguagem C:

**Exemplo com Matriz:**



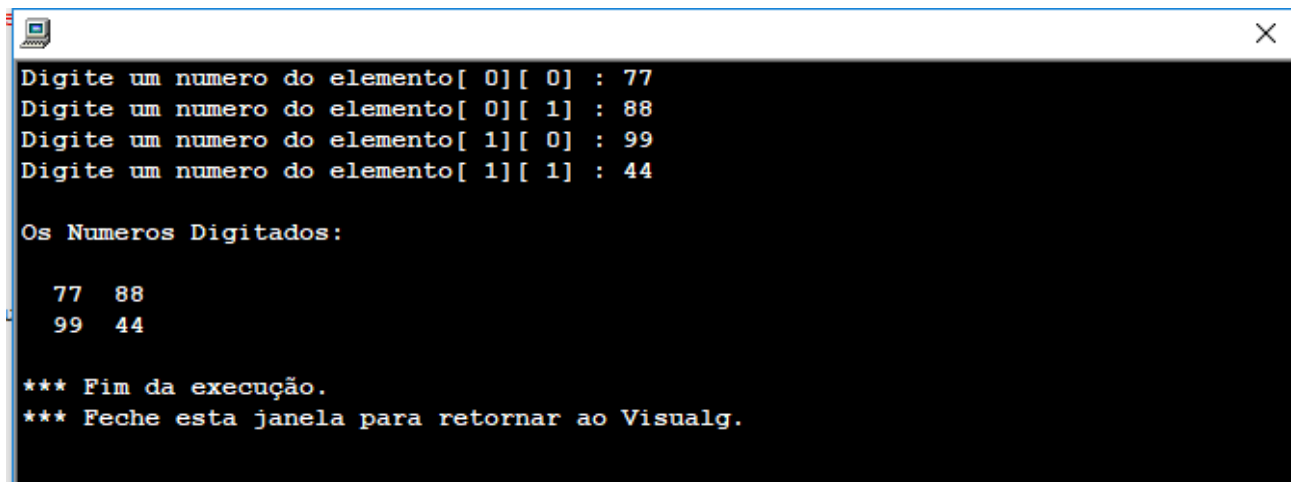
Considere um algoritmo que vai pegar quatro números digitados pelo usuário, colocá-los em forma de matriz e imprimi-los na tela.

### 1) Pseudocódigo

```
algoritmo "exemploMatriz"
var
matriz: vetor[0..1,0..1] de inteiro
linha, coluna: inteiro
inicio
para linha de 0 ate 1 faca
para coluna de 0 ate 1 faca
escreva("Digite um numero do elemento[" ,linha,"][",coluna,"] : ")
leia(matriz[linha,coluna])
fimpara
fimpara
escreval(" ")
escreval("Os Numeros Digitados:")
escreval(" ")
para linha de 0 ate 1 faca
para coluna de 0 ate 1 faca
escreva(" ",matriz[linha,coluna]:2)
fimpara
escreval(" ")
fimpara
finalgoritmo
```

A Figura 9 mostra a saída do exemplo anterior:

Figura 9 – Saída do algoritmo exemploMatriz



```
Digite um numero do elemento[ 0][ 0] : 77
Digite um numero do elemento[ 0][ 1] : 88
Digite um numero do elemento[ 1][ 0] : 99
Digite um numero do elemento[ 1][ 1] : 44

Os Numeros Digitados:

 77  88
 99  44

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

## 2) Linguagem C

```
# include <stdio.h>

int main()
{
    int linha, coluna, matriz[2][2];
    for (linha=0; linha<2; ++linha)
        for (coluna=0; coluna<2; ++coluna){
            printf("Digite um numero do elemento[%d][%d] : ", linha, coluna);
            scanf("%d", &matriz[ linha ][ coluna ]);
        }
    printf("\nOs Numeros Digitados: \n");
    printf ("\n");
    for (linha=0; linha<2; ++linha)
    {
        for (coluna=0; coluna<2; ++coluna)
            printf ("%d ", matriz[linha][coluna]);
        printf ("\n");
    }
    printf ("\n");
    system("pause");
    return(0);
}
```

A Figura 10 mostra a saída do exemplo anterior:



Figura 10 – Saída do algoritmo em Linguagem C

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\Matriz\bin\Debug\Matriz.exe"
Digite um numero do elemento[0][0] : 77
Digite um numero do elemento[0][1] : 88
Digite um numero do elemento[1][0] : 99
Digite um numero do elemento[1][1] : 44

Os Numeros Digitados:

77 88
99 44

Pressione qualquer tecla para continuar. . . _
```

## TEMA 4 – CADEIA DE CARACTERES

Na ciência da computação, programação *string* é um tipo de variável usada para armazenar cadeia de caracteres com conteúdo que pode ser alterado ou substituído por outros elementos para formar uma nova cadeia de caracteres (Mizrahi, 2008; Ascencio, 2012; Puga; Rissetti, 2016).

Para escrever um texto, é necessário representar essa cadeia de caracteres. Por isso, usa-se a palavra-chave "**string**", que é vista como sendo um tipo de dado que armazena os elementos da cadeia em uma sequência na memória, utilizando alguma codificação preestabelecida.

A linguagem C não possui um tipo de dado similar à **string**. Em vez disso, para armazenar uma cadeia de caracteres, utiliza vetores (matrizes unidimensionais) da seguinte forma:

- O tipo **char** armazena o índice da tabela ASCII correspondente ao caractere.
- Para definir uma **string** em C, é necessário definir um vetor com o número máximo de caracteres da cadeia entre colchetes, **[NúmeroDeElementos]**, que será igual ao número de elementos da tabela, menos um que foi reservado para o fim da cadeia com o valor zero ('0').
- A sintaxe de um vetor em C para representar uma **string** começa com a definição dos dados que serão armazenados, do tipo "**char**", seguido pelo nome do vetor "**NomeDaString**" e os colchetes que define a



quantidade de elementos que será armazenado na memória “[NúmeroDeElementos]”.

- char NomeDaString[NúmeroDeElementos]
- char nome[8];

Na linguagem C, a estratégia para armazenar caracteres foi usar um array (vetor). Os arrays já representam um conjunto de dados relacionados, que são acessados por um índice. Portanto, a palavra "UNINTER", em na linguagem de programação C, é um array que contém oito [8] posições, conforme mostrado a seguir:

1) char nome [8] = { 'U', 'N', 'I', 'N', 'T', 'E', 'R', '\0' };

Segundo Mizrahi (2008), o uso mais importante do vetor é aplicado à criação de tipos de dados para armazenar e manipular textos, palavras, nomes e sentenças.

Em outras palavras, cada texto, palavra, nome e sentença é um conjunto de caracteres, em que cada um ocupa um byte de memória, armazenado em sequência e terminado por um byte de valor zero ('0'). Cada caractere é um elemento independente no vetor e pode ser acessado por meio de um índice (Mizrahi, 2008).

## TEMA 5 – INICIALIZAR UMA CADEIA DE CARACTERES

Uma *string* pode ser inicializada automaticamente pelo programa ou pode receber um valor por meio do teclado. Na linguagem C, deve-se inicializar a *string*, ou seja, preencher os espaços em branco da tabela, com caracteres, sabendo que ele vai necessariamente conter o caractere final da cadeia '\0' (Mizrahi, 2008; Ascencio, 2012).

Usa-se as duas formas, mostradas a seguir, para inicializar uma *string* no momento da sua declaração na linguagem de programação C:

- 1) char nome [ ] = "UNINTER";
- 2) char nome [ ] = { 'U', 'N', 'I', 'N', 'T', 'E', 'R', '\0' };

No primeiro caso, a variável nome recebeu as letras separadamente (inclusive o caractere nulo (0)). Por isso, cada uma das letras estava entre ( ' ' ) – esta é a maneira de identificar um caractere isoladamente. Já no segundo caso, a variável nome foi inicializada com uma palavra, recebendo



automaticamente o caractere nulo. Por isso, a palavra UNINTER estava entre ("") – esta é a maneira de identificar uma cadeia de caracteres (Ascencio, 2012).

Em ambos os casos, não precisou expressar o número de posições dentro dos colchetes, na inicialização o número foi definido automaticamente na inicialização da *string*.

A seguir, veremos um exemplo de algoritmo usando *string* em pseudocódigo e Linguagem C:

### **Exemplo com *String*:**

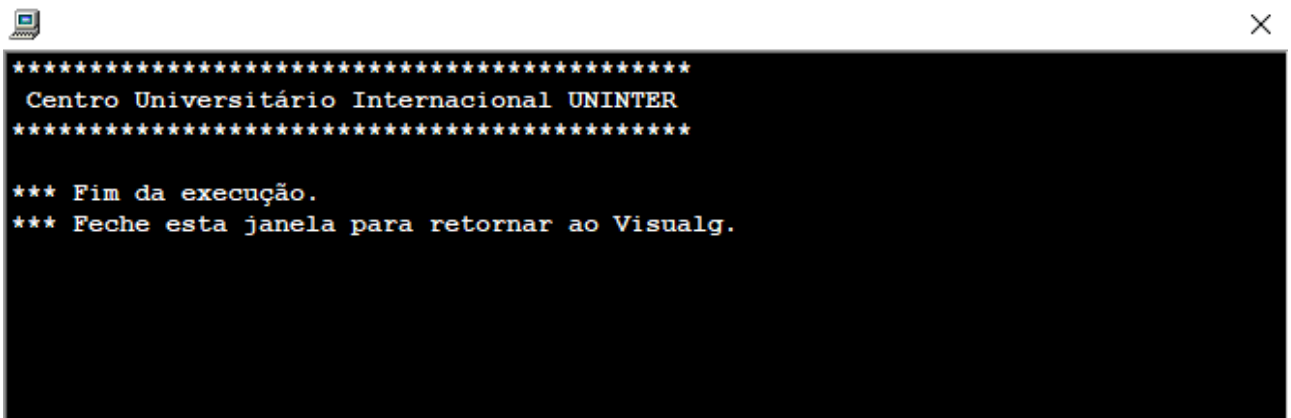
Considere um algoritmo que vai pegar quatro conjuntos de caracteres e imprimi-los na tela do usuário.

#### **1) Pseudocódigo**

```
algoritmo "exemploString"
var
nome1 : caractere
nome2 : caractere
nome3 : caractere
asteriscos : caractere
inicio
nome1 <- "Centro Universitário"
nome2 <- "Internacional"
nome3 <- "UNINTER"
asteriscos <- "*****"
escreval(asteriscos)
escreval(" ",nome1," ",nome2," ",nome3)
escreval(asteriscos)
finalgoritmo
```

Repare que a sintaxe do pseudocódigo reconhece a cadeia de caracteres sem precisar definir um vetor[ ]. Para o reconhecimento automático da cadeia de caracteres, basta declarar a variável como do tipo **caractere**. Nesse exemplo, criamos quatro variáveis para armazenar quatro cadeias de caracteres e imprimi-los na tela do usuário. Teremos o resultado mostrado na Figura 11.

Figura 11 – Saída do algoritmo exemploString em pseudocódigo



```
*****
Centro Universitário Internacional UNINTER
*****

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

## 2) Linguagem C

```
#include<stdio.h>
#include<conio.h>
#include <locale.h>
int main()
{
    char nome1[ ]= "Centro Universitário";
    char nome2[ ]= "Internacional";
    char nome3[ ]= "UNINTER";
    char asteriscos[]= "\n*****\n";
    printf("%s",asteriscos);
    printf(" %s %s %s", nome1,nome2,nome3, setlocale(LC_ALL,""));
    printf("%s",asteriscos);
    system("pause");
    return 0;
}
```

Para o mesmo exemplo na linguagem C definimos quatro vetores do tipo char para armazenar as cadeias de caracteres. A função printf() não reconhece acentuação da língua portuguesa (pt-br); para resolver esse problema usamos função setlocale() da biblioteca locale.h, que vai usar o idioma padrão do sistema operacional. Como saída para esse algoritmo, teremos o resultado mostrado na Figura 12.



Figura 12 – Saída do algoritmo em linguagem de programação C2

```
"C:\Users\Casa\Documents\Sandro\FACULDADES\UNINTER\Linguagem de Programação\String\bin\Debug\String.exe"

*****
Centro Universitário Internacional UNINTER
*****
Pressione qualquer tecla para continuar. . . _
```

## FINALIZANDO

Nesta aula, aprendemos os principais conceitos que envolvem os principais conceitos e aplicações de vetor, matriz e cadeia de caracteres. Também tivemos uma introdução de como declará-los e inicializá-los nas construções de algoritmos em pseudocódigos e linguagem de programação C. Portanto, aproveite a disciplina e bons estudos.



## REFERÊNCIAS

ASCENCIO, A. F. G. **Fundamentos da Programação de Computadores:** Algoritmos, Pascal, C/C++ (padrão ANSI) JAVA. 3. ed. São Paulo: Pearson, 2012.

MIZRAHI, V. V. **Treinamento em Linguagem C.** 2. Edição. São Paulo: Pearson, 2008.

PUGA, S.; RISSETTI, G. **Lógica de programação e estruturas de dados.** Pearson Educación, 2016.