

Algoritmos e Programação I

Linguagem de programação C

Trabalhando com Arrays

Colecionando dados.

Definição geral

- Uma matriz é uma coleção de **dados** do mesmo **tipo** que é referenciada por um **nome** comum;
- Seus elementos são acessados por meio de um **índice**;
- O **endereço** mais baixo desse **índice** corresponde ao primeiro elemento e o mais alto ao último;
- As matrizes ocupam **posições contíguas** na memória;
- As “**strings**” são o tipo de matriz mais comum em C;
- Matrizes podem ser:
 - **Unidimensionais** ou;
 - **Multidimensionais**.

Vetores

Arrays Unidimensionais

Unidimensional ou Vetor

- A declaração de um vetor é dada da seguinte forma:

tipo nome[tamanho]

- Onde:
 - **tipo** define o tipo de dado que será armazenado;
 - **nome** é o nome da variável que armazenará os dados;
 - **[tamanho]** entre colchetes vem o tamanho do vetor ou número de elementos que o mesmo poderá conter.
- Como qualquer variável, as matrizes **devem ser declaradas** antes da sua utilização;
- Em C, o índice de uma matriz inicia-se em **0 (zero)** . Então, se declararmos uma matriz de 10 elementos, teremos um índice de 0 a 9.

Declarando um vetor unidimensional

- A declaração de um vetor é feita da seguinte forma:

```
main()
{
    int notas[5];
}
```

Declarando um vetor unidimensional

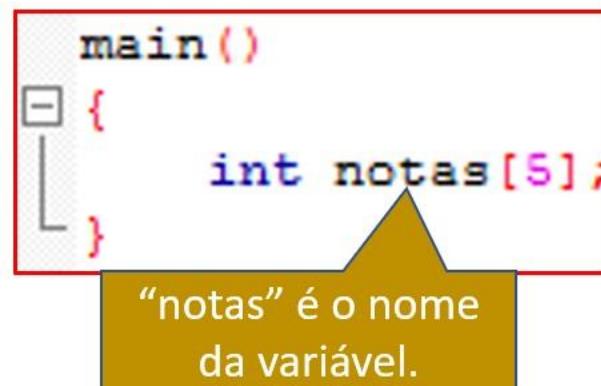
- A declaração de um vetor é feita da seguinte forma:

The diagram shows a code snippet for declaring a one-dimensional array. It consists of three parts: a blue speech bubble containing the text "Tipo de dado 'inteiro'.", a red rectangular box containing the code "int notas[5];", and a small icon of a computer monitor in the bottom-left corner.

```
int notas[5];
```

Declarando um vetor unidimensional

- A declaração de um vetor é feita da seguinte forma:



Declarando um vetor unidimensional

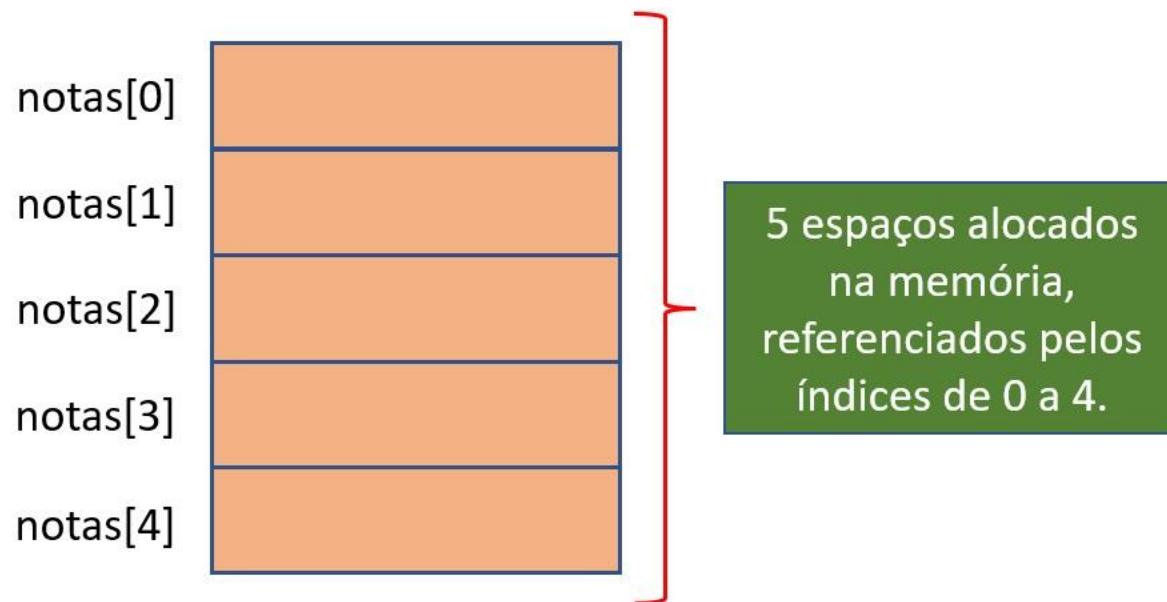
- A declaração de um vetor é feita da seguinte forma:

```
main()
{
    int notas[5];
```

5 é o tamanho do vetor.

Declarando um vetor

- Na memória teremos:



Referenciando um elemento

- Os elementos são referenciados pelo **seu número de índice**, que indica a sua **posição dentro do vetor**.
- No exemplo a seguir, utilizamos um laço “**for**” para preencher os elementos do vetor:

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

    for(i = 0;i < 5;i++)
    {
        printf("Digite a nota do aluno %d ",i);
        scanf("%d",&notas[i]);
    }
}
```

Referenciando um elemento

- Os elementos são referenciados pelo **seu número de índice**, que indica a sua **posição dentro do vetor**.
- No exemplo a seguir, utilizamos um laço “**for**” para preencher os elementos do vetor:

```
main
{
    Laço para
    preenchimento
    do vetor.

    for(i = 0;i < 5;i++)
    {
        printf("Digite a nota do aluno %d ",i);
        scanf("%d",&notas[i]);
    }
}
```

Referenciando um elemento

- Os elementos são referenciados pelo **seu número de índice**, que indica a sua **posição dentro do vetor**.
- No exemplo a seguir, utilizamos um laço “**for**” para preencher os elementos do vetor:

```
main()
{
    int notas[5];
    int i, soma;
    for(i = 0; i <
    {
        printf("Digite a nota do aluno %d ", i);
        scanf("%d", &notas[i]);
    }
}
```

O índice vai variar de 0 a 4, permitindo assim que as notas sejam inseridas no vetor em posições diferentes.

Referenciando um elemento

- Na memória teremos:



Inicializando um vetor unidimensional

- Como uma variável comum, os vetores podem ser inicializados já durante a declaração;
- Exemplo:

```
int contador[5] = {0, 0, 0, 0, 0};
```

```
Int inteiros[5] = {0, 1, 2, 3, 4};
```

- Vetores do tipo caractere são inicializados de maneira similar:

```
char vogais[5] = {'a', 'e', 'i', 'o', 'u'};
```

- Um vetor não precisa ser inicializado por completo:

```
float dados[500] = {20.5, 10, 111.7};
```

- Os três primeiros elementos foram declarados, o restante será preenchido com zeros.

- Uma posição de um vetor pode ser inicializada especificamente:

```
float dados[500] = { [2] = 500.5, [1] = 300.0, [0] = 100.0 };
```

- A inicialização também pode ser através de uma expressão:

```
int x = 123;
```

```
int a[10] = {[9] = x + 1, [2] = 2, [1] = 0, [0] = 10};
```

Exemplo 01 – Inicializando um vetor

```
1  #include <stdio.h>
2  int main (void)
3  {
4      int array_valores[10] = { 0, 1, 4, 9, 16 };
5      int i;
6
7      for ( i = 5; i < 10; ++i )
8          array_valores[i] = i * i;
9
10     for ( i = 0; i < 10; ++i )
11         printf ("array_valores[%i] = %i\n", i, array_valores[i]);
12
13     return 0;
14 }
```

Exemplo 01 – Inicializando um vetor

```
1  Vetor com 10 posições. >
2
3  {
4      int array_valores[10] = { 0, 1, 4, 9, 16 };
5      int i;
6
7      for ( i = 5; i < 10; ++i )
8          array_valores[i] = i * i;
9
10     for ( i = 0; i < 10; ++i )
11         printf ("array_valores[%i] = %i\n", i, array_valores[i]);
12
13     return 0;
14 }
```

Exemplo 01 – Inicializando um vetor

```
1  #include <stdio.h>
2  int main (void)
3  {
4      int array_valores[10] = { 0, 1, 4, 9, 16 };
5      int i;
6
7      for ( i = 5; i < 10; ++i )
8          array_valores[i] = i * i;
9
10     for ( i = 0; i < 10; ++i )
11         printf ("array_valores[%i] = %i\n", i, array_valores[i]);
12
13     return 0;
14 }
```

As 5 primeiras posições são
inicializadas na declaração.

Exemplo 01 – Inicializando um vetor

```
1 #include <stdio.h>
2 int main (void)
3 {
4     int array_valores[10];
5     array_valores[0] = { 0, 1, 4, 9, 16 };
6
7     for ( i = 5; i < 10; ++i )
8         array_valores[i] = i * i;
9
10    for ( i = 0; i < 10; ++i )
11        printf ("array_valores[%i] = %i\n", i, array_valores[i]);
12
13    return 0;
14 }
```

Laço “for” para inicialização dos outros elementos do vetor.

Exemplo 01 – Inicializando um vetor

```
1  #include <stdio.h>
2  int main (void)
3  {
4      int array_valores[10] = { 0, 1, 4, 9, 16 };
5      int i;
6
7      for ( i = 5; i < 10; ++i )
8          array_valores[i] = i * i;
9
10     for ( i = 0; i < 10; ++i )
11         printf ("array_valores[%d] = %d\n", i, array_valores[i]);
12
13     return 0;
14 }
```

Saída do programa.

```
array_valores[0] = 0
array_valores[1] = 1
array_valores[2] = 4
array_valores[3] = 9
array_valores[4] = 16
array_valores[5] = 25
array_valores[6] = 36
array_valores[7] = 49
array_valores[8] = 64
array_valores[9] = 81

Process returned 0 <0x0>    execution time : 0.245 s
Press any key to continue.
```

Exemplo 02 – Média de notas

```
7     main()
8     {
9         int notas[5];
10        int i, soma;
11
12        for(i = 0;i < 5;i++)
13        {
14            printf("Digite a nota do aluno %d ",i);
15            scanf("%d",&notas[i]);
16        }
17
18        soma = 0;
19
20        for(i = 0;i < 5;i++)
21            soma = soma + notas[i];
22
23        printf("\nMedia das notas: %d:",soma / 5);
24    }
```

Exemplo 02 – Média de notas

```
7     main()
8     {
9         int notas[5];
10        int i, soma;
11
12        for(i = 0; i < 5; i++)
13        {
14            printf("Digite a nota do aluno %d ", i);
15            scanf("%d", &notas[i]);
16        }
17
18        soma = 0;
19
20        for(i = 0; i < 5; i++)
21            soma = soma + notas[i];
22
23        printf("\nMedia das notas: %d:", soma / 5);
24    }
```

Entrada de
notas no vetor,
posição "i".

Exemplo 02 – Média de notas

```
7     main()
8     {
9         int notas[5];
10        int i, soma;
11
12        for(i = 0;i < 5;i++)
13        {
14            printf("Digite a nota do aluno %d ",i);
15            scanf("%d",&notas[i]);
16
17            Variável "soma"
18            recebe o somatório
19            das notas.
20
21            for(i = 0;i < 5;i++)
22                soma = soma + notas[i];
23
24            printf("\nMedia das notas: %d:",soma / 5);
25        }
```

Exemplo 02 – Média de notas

```
7     main()
8     {
9         int notas[5];
10        int i, soma;
11
12        for(i = 0; i < 5; i++)
13        {
14            printf("Digite a nota do aluno %d ", i);
15            scanf("%d", &notas[i]);
16        }
17
18
19        Exibe o resultado
20        da média das
21        notas.
22
23        printf("\nMédia das notas: %d:", soma / 5);
24    }
```

Inicializando um vetor unidimensional

- A linguagem C permite que um vetor seja declarado sem que determinemos de imediato seu tamanho. Veja o exemplo:

```
1  #include <stdio.h>
2  int main (void)
3  {
4      char palavra[] = { 'F', 'e', 'l', 'i', 'z', '!' };
5      int i;
6
7      for ( i = 0; i < 6; ++i )
8          printf ("%c", palavra[i]);
9
10     printf ("\n");
11
12     return 0;
13 }
```

Inicializando um vetor unidimensional

- A linguagem C permite que um vetor seja declarado sem que determinemos de imediato seu tamanho. Veja o exemplo:

Declaração do vetor.

```
#include <stdio.h>
int main()
{
    char palavra[] = { 'F', 'e', 'l', 'i', 'z', '!' };
    int i;

    for ( i = 0; i < 6; ++i )
        printf ("%c", palavra[i]);

    printf ("\n");

    return 0;
}
```

Inicializando um vetor unidimensional

- A linguagem C permite que um vetor seja declarado sem que determinemos de imediato seu tamanho. Veja o exemplo:

```
1 #include <stdio.h>
2 int main (void)
3 {
4     char palavra[] = { 'F', 'e', 'l', 'i', 'z', '!' };
5     int i;
6
7     for ( i = 0; i < 6; ++i )
8         printf ("%c", palavra[i]);
9
10    printf ("\n");
11
12    return 0;
13 }
```

Seu tamanho será determinado pelo maior índice inicializado.

Inicializando um vetor unidimensional

- A linguagem C permite que um vetor seja declarado sem que determinemos de imediato seu tamanho. Veja o exemplo:

```
1  #include <stdio.h>
2  int main (void)
3  {
4      O laço vai até o tamanho do índice
5          do vetor, que é 6.
6
7      for ( i = 0; i < 6; ++i )
8          printf ("%c", palavra[i]);
9
10     printf ("\n");
11
12     return 0;
13 }
```

Inicializando um vetor unidimensional

- A linguagem C permite que um vetor seja declarado sem que determinemos de imediato seu tamanho. Veja o exemplo:

```
1  #include <stdio.h>
2  int main (void)
3  {
4      char palavra[] = { 'F', 'e', 'l', 'i', 'z', '!' };
5      int i;
6
7      for ( i = 0; i < 6; ++i )
8          printf ("%c", pa
9
10     printf ("\n");
11
12     return 0;
13 }
```

Saída do programa.

```
Feliz!
Process returned 0 (0x0)  execution time : 0.005 s
Press any key to continue.
```

Inicializando um vetor unidimensional

- Outro exemplo de inicialização de vetores:

```
float dados[] = { [0] = 1.0, [49] = 100.0, [99] = 200.0 };
```

- No exemplo acima o tamanho do vetor é 100, pois 99 é o maior índice declarado no vetor.

Exemplo 03 – uso de vetores inteiros e de caracteres

- A tarefa é a de desenvolver um programa que converte um número inteiro positivo de sua representação base 10 na sua representação equivalente em outra base.
- Como entradas para o programa, você especifica o número a ser convertido e também a base para o qual deseja que o número seja convertido.
- O primeiro passo no desenvolvimento de tal programa é conceber um algoritmo para converter um número de base 10 para outra base.
- Um algoritmo para gerar os dígitos do número convertido pode ser informalmente indicado como se segue:
 - Um dígito do número convertido é obtido tomando o **modulo** do número pelo **número da base**;
 - O número é então **dividido pela base**, com qualquer resto fracionário descartado, e o processo é repetido até que o número **chegar a zero**.

Exemplo 03 – uso de vetores inteiros e de caracteres

Número	Número % 2	Número / 2
10	0	5
5	1	2
2	0	1
1	1	0

Exemplo 03 – uso de vetores inteiros e de caracteres

- Para escrever um programa que executa o processo de conversão anterior, você deve levar algumas coisas em consideração:
 - Em primeiro lugar, o fato do algoritmo gerar os dígitos do número convertido em ordem inversa. Não podemos esperar que o usuário leia o resultado da direita para a esquerda, ou a partir de baixo da página para cima. Portanto, você deve corrigir esse problema.
 - Ao invés de simplesmente exibir cada dígito como ele é gerado, você pode fazer o programa armazenar cada dígito dentro de uma matriz. Então, quando você terminar de converter o número, você pode exibir o conteúdo do array na ordem correta.
 - Em segundo lugar, você deve perceber que o programa deve ser especificado para lidar com a conversão de números em bases até 16. Isto significa que quaisquer dígitos do número convertido que estão entre 10 e 15 deve ser exibido usando as letras correspondentes, de A a F. É ai que nossa matriz de caracteres entra em cena.
 - Examine o a seguir para ver como estas duas questões são tratadas. Este programa também introduz o qualificador de tipo **const**, que é usado para as variáveis cujo valor não mudam durante a execução do programa.

Exemplo 03 – uso de vetores inteiros e de caracteres

```
3 #include <stdio.h>
4 int main (void)
5 {
6     const char baseDigitos[16] = {
7         '0', '1', '2', '3', '4', '5', '6', '7',
8         '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };
9     int convertidoNum[64];
10    long int numAConverter;
11    int proxDigit, base, indice = 0;
12
13 //Recebe o número e a base.
14
15    printf ("Numero para ser convertido? ");
16    scanf ("%ld", &numAConverter);
17    printf ("Base? ");
18    scanf ("%i", &base);
19
```

Exemplo 03 – uso de vetores inteiros e de caracteres

Define estes valores como
“constantes” dentro do programa.

```
3 // Criei um vetor com os dígitos da base 16
4
5 {
6     const char baseDigitos[16] = {
7         '0', '1', '2', '3', '4', '5', '6', '7',
8         '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };
9
10    int convertidoNum[64];
11    long int numAConverter;
12    int proxDigit, base, indice = 0;
13
14    //Recebe o número e a base.
15
16    printf ("Número para ser convertido? ");
17    scanf ("%ld", &numAConverter);
18    printf ("Base? ");
19    scanf ("%i", &base);
```

Exemplo 03 – uso de vetores inteiros e de caracteres

```
3 #include <stdio.h>
4 int main
5 {
6     const
7         int convertidoNum[64];
8         long int numAConverter;
9         int proxDigit, base, indice = 0;
10
11     //Recebe o número e a base.
12
13     printf ("Numero para ser convertido? ");
14     scanf ("%ld", &numAConverter);
15     printf ("Base? ");
16     scanf ("%i", &base);
17 }
```

Vetor para receber o número convertido.

Exemplo 03 – uso de vetores inteiros e de caracteres

```
20 // Converte para a base indicada.
21
22 do {
23     convertidoNum[indice] = numAConverter % base;
24     ++indice;
25     numAConverter = numAConverter / base;
26 }
27 while ( numAConverter != 0 );
28
29 // Exibe o resultado em ordem reversa
30
31 printf ("Numero convertido = ");
32 for (--indice; indice >= 0; --indice ) {
33     proxDigit = convertidoNum[indice];
34     printf ("%c", baseDigitos[proxDigit]);
35 }
36
37 printf ("\n");
38
39 return 0;
40 }
```

Exemplo 03 – uso de vetores inteiros e de caracteres

```
20 // Converte para a base indicada.  
21  
22     do {  
23         convertidoNum[indice] = numAConverter % base;  
24         ++indice;  
25         numAConverter = numAConverter / base;  
26     }  
27     while ( numAConverter != 0 );  
28  
29 // Exibe o resultado em ordem reversa  
30  
31     printf ("Numero convertido = ");  
32     for (--indice; indice >= 0; --indice ) {  
33         proxDigit = convertidoNum[indice];  
34         printf ("%c", baseDigitos[proxDigit]);  
35     }  
36  
37     printf ("\n");  
38  
39     return 0;  
40 }
```

Rotina para converter o número para a base indicada.

Exemplo 03 – uso de vetores inteiros e de caracteres

```
20 // Converte para a base indicada.  
21  
22 do {  
23     convertidoNum[indice] = numAConverter % base;  
24     ++indice;  
25     numAConverter = numAConverter / base;  
26 }  
27 while ( numAConverter != 0 );  
28  
29 // Exibe o resultado em ordem reversa  
30  
31 printf ("Numero convertido = ");  
32 for (--indice; indice >= 0; --indice ) {  
33     proxDigit = convertidoNum[indice];  
34     printf ("%c", baseDigitos[proxDigit]);  
35 }  
36  
37 printf ("\n");  
38  
39 return 0;  
40 }
```

Rotina para exibir o número na ordem correta.

Diretiva #define

- Observe o seguinte programa:

```
4  #include <stdio.h>
5
6  #define LIM 40
7
8  main()
9  {
10     float notas[LIM], soma = 0.0;
11     int i = 0;
12
13     do
14     {
15         printf("Digite a nota do aluno %d: ", i);
16         scanf("%f", &notas[i]);
17         if(notas[i] > 0)
18             soma += notas[i];
19     }while(notas[i++] > 0);
20
21     printf("Media das notas: %.2f", soma / (i - 1));
22 }
```

Diretiva #define

- Na linha 6 do programa anterior, temos a seguinte definição: `#define LIM 6`. Isso faz com que o programa substitua a constante LIM por **6** onde a mesma for encontrada.
- Algumas diferenças entre o uso de **#define** e **const**:
 - **#define** é um pré-processador , por isso deve ser definido no cabeçalho do programa;
 - Uma constante definida por **#define**, não existe fisicamente na memória;
 - **const** é uma palavra reservada em C;

Exemplo #define

```
1  #include <stdio.h>
2  #define PI 3.1416
3  #define VERSAO "1.0"
4  int main ()
5  {
6      printf ("Programa versao %s\n", VERSAO);
7      printf ("O numero pi vale: %f\n", PI);
8      return 0;
9 }
```

Matrizes

Arrays Multidimensionais

Matriz Multidimensional

- É uma **matriz de matrizes**, pois os elementos de uma matriz multidimensional são **matrizes de uma dimensão**;

- A declaração pode ser feita da seguinte forma:

tipo nome[indice_1][indice_2];

- A matriz bidimensional, duas dimensões, é a mais comum;

- Onde:

- **tipo** é o tipo de dado armazenado na matriz;

- **nome** é o nome identificador da matriz;

- **[indice_1][indice_2]...[indice_n]** são os índices da matriz. O número de elementos é conseguido multiplicando-se os índices da matriz.

Exemplo 01

```
9     main()
10    {
11        int t, i, num[3][4];
12
13        // Preenchendo a matriz com elementos
14        for(t = 0; t < 3; ++t)
15            for(i = 0; i < 4; ++i)
16                num[t][i] = (t * 4) + i + 1;
17
18        // Exibindo os elementos da matriz.
19        for(t = 0; t < 3; ++t)
20            for(i = 0; i < 4; ++i)
21                printf("%3d ", num[t][i]);
22            printf("\n");
23    }
```

Exemplo 01

```
9     main()
10    {
11        int t, i, num[3][4];
12
13        // Preenchendo a matriz com elementos
14        for(t = 0; t < 3; ++t)
15            for(i = 0; i < 4; ++i)
16                num[t][i] = (t * 4) + i + 1;
17
18        // Exibindo os elementos da matriz.
19        for(t = 0; t < 3; ++t)
20            for(i = 0; i < 4; ++i)
21                printf("%3d ", num[t][i]);
22            printf("\n");
23    }
```

Matriz bidimensional do tipo
“inteiro” de nome “num”
com 12 elementos (3x4).

Exemplo 01

```
9     main()
10    {
11        int t, i, num[3][4];
12
13        // Preenchendo a matriz com valores
14        for(t = 0; t < 3; ++t)
15            for(i = 0; i < 4; ++i)
16                num[t][i] = (t * 4) + i + 1;
17
18        // Exibindo os elementos da matriz.
19        for(t = 0; t < 3; ++t)
20            for(i = 0; i < 4; ++i)
21                printf("%3d ", num[t][i]);
22            printf("\n");
23    }
```

A matriz possui 03 linhas e 04 colunas ([3][4])

Exemplo 01

```
9     main()
10    {
11        int t, i, num[3][4];
12
13        // Preenchendo a matriz com elementos
14        for(t = 0; t < 3; ++t)
15            for(i = 0; i < 4; ++i)
16                num[t][i] = (t * 4) + i + 1;
17
18        // Exibindo os elementos da matriz.
19        for(t = 0; t < 3; ++t)
20            for(i = 0; i < 4; ++i)
21                printf("%3d ", num[t][i]);
22            printf("\n");
23    }
```

Laço para variar a posição das linhas da matriz, "t".

Exemplo 01

```
9     main()
10    {
11        int t, i, num[3][4];
12
13        // Preenchendo a matriz com elementos
14        for(t = 0; t < 3; ++t)
15            for(i = 0; i < 4; ++i)
16                num[t][i] = (t * 4) + i + 1;
17
18    Laço para variar a posição das
19    colunas da matriz, "i".
20
21    os elementos da matriz.
22
23 }
```

Exemplo 01

```
9     main()
10    {
11        int t, i, num[3][4];
12
13        // Preenchendo a matriz com elementos
14        for(t = 0; t < 3; ++t)
15            for(i = 0; i < 4; ++i)
16                num[t][i] = (t * 4) + i + 1;
17
18        elemento [t][i] da matriz      elementos da matriz.
19        ; ++t)
20            for(i = 0; i < 4; ++i)
21                printf("%3d ", num[t][i]);
22            printf("\n");
23    }
```

Exemplo 01

```
9     main()
10    {
11        int t, i, num[3][4];
12
13        // Preenchendo a matriz com elementos
14        for(t = 0; t < 3; ++t)
15            for(i = 0; i < 4; ++i)
16                num[t][i] = (t * 4) + i + 1;
17
18        // Exibi Expressão para preencher a iz.
19        for(t =
20            for(i = 0; i < 4; ++i)
21                printf("%3d ", num[t][i]);
22                printf("\n");
23    }
```

Expressão para preencher a iz.
posição [t][i] com seu elemento.

Exemplo 01

```
9     main()
10    {
11        int t, i, num[3][4];
12
13        // Preenchendo a matriz com elementos
14        for(t = 0; t < 3; ++t)
15            for(i = 0; i < 4; ++i)
16                num[t][i] = (t * 4) + i + 1;
17
18        // Exibindo os elementos da matriz.
19        for(t = 0; t < 3; ++t)
20            for(i = 0; i < 4; ++i)
21                printf("%3d ", num[t][i]);
22            printf("\n");
23    }
```

Imprime os elementos da matriz.

Exemplo 01

```
9     main()
10    {
11        int t, i, num[3][4];
12
13        // Preenchendo a matriz com elementos
14        for(t = 0; t < 3; ++t)
15            for(i = 0; i < 4; ++i)
16                num[t][i] = Saída em tela.
17
18        // Exibindo os elementos da matriz
19
20        1 2 3 4 5 6 7 8 9 10 11 12
21
22 Process returned 10 (0xA) execution time : 0.011 s
23 Press any key to continue.
```

Inicialização de matrizes multidimensionais

- Matrizes bidimensionais podem ser inicializadas de forma análoga aos seus homólogos unidimensionais;
- Elementos quando listados para a inicialização, os valores são listados por linha.
- Pares de chaves são utilizados para separar a lista de inicializadores para uma linha, linha por linha;
- Assim, podemos definir os elementos da seguinte forma:

```
int M[4][5] = {  
    { 10, 5, -3, 17, 82 },  
    { 9, 0, 0, 8, -7 },  
    { 32, 20, 1, 0, 14 },  
    { 0, 0, 8, 7, 6 }  
};
```

Inicialização de matrizes multidimensionais

- Preste especial atenção à sintaxe da declaração anterior;
- Note-se que **vírgulas são necessárias** depois de cada chave que fecha uma linha, **exceto** no caso da linha final;
- A utilização dos **pares de chaves** é realmente **opcional**. Assim, a declaração anterior também poderia ter sido escrita como segue:

```
int M[4][5] = { 10, 5, -3, 17, 82, 9, 0, 0, 8, -7, 32,  
                20, 1, 0, 14, 0, 0, 8, 7, 6 };
```

Inicialização de matrizes multidimensionais

- Tal como acontece com matrizes unidimensionais, não é necessário que toda a matriz seja inicializada.
- Uma declaração como:

```
int M[4][5] = {  
    { 10, 5, -3 },  
    { 9, 0, 0 },  
    { 32, 20, 1 },  
    { 0, 0, 8 }  
};
```

- inicializa apenas os primeiros três elementos de cada linha da matriz com os valores indicados. Os demais valores são definidos como 0.
- Note-se que, neste caso, os pares de chaves interiores são necessários para forçar a inicialização correta. Sem elas, as duas primeiras linhas e os primeiros dois elementos da linha 3 teriam sido inicializados em vez disso.