MATRIZ PARA ARMAZENAMENTO DE DADOS

Matrizes

Neste vídeo, vamos explorar as matrizes em C e aprender como manipular eficientemente dados multidimensionais. Aprenderemos sobre a declaração, inicialização e manipulação de matrizes, incluindo operações como acesso a elementos, atribuição, cálculos e iteração.

Comparação com o vetor

A **matriz** é uma estrutura de dados homogênea, tal qual o vetor. Por isso, todos os seus elementos são do mesmo tipo. A diferença fundamental da **matriz** para o vetor é a quantidade de índices que são usados para acessar um elemento:

Matriz

Na matriz, o acesso a cada elemento ocorre pelo uso de dois índices.

Vetor

No vetor, o elemento é acessado pelo índice, que representa a posição relativa desse elemento no vetor.

Quando estudamos vetores, vimos que, para representar 3 notas de 50 alunos de uma turma, foi preciso usar 3 vetores de elementos do tipo real, conforme a declaração a seguir:

float prova1[50], prova2[50], prova3[50];

Vamos observar como que ficam as cadeias dos três vetores a seguir:

0	1	2	3	.....	47	48	49
7.00	8.50	9.35	8.45	.....	9.00	7.90	9.60

Cadeia do vetor prova 1.

0	1	2	3	.....	47	48	49
9.70	9.20	8.95	9.00	.....	10.00	6.40	4.30

Cadeia do vetor prova 2.

0	1	2	3	.....	47	48	49
10.00	8.50	7.50	8.80	.....	9.20	8.00	9.00

Cadeia do vetor prova 3.

Com essa estrutura, armazenamos em cada vetor os dados de 1 prova (são 3). Convencionamos que cada aluno tem suas notas no mesmo índice de cada um dos 3 vetores.

Com isso, o aluno 1 tem suas notas na posição 0 (zero) de cada vetor, o aluno 2, na posição 1, e assim por diante, até que, na posição de cada vetor, temos as notas do último aluno.

À medida que aumenta o número de provas, precisamos de mais vetores. Se fossem 10 provas, seriam necessários 10 vetores de 50 elementos do tipo *float* (real). Se fossem 20 provas, seriam necessários 20 vetores, e assim sucessivamente. Manipular 3 vetores pode ser até factível, porém, com mais de 20, começa a complicar.

A matriz é a estrutura de dados ideal para esse tipo de situação.

Em vez de N vetores, declaramos e usamos apenas 1 matriz. É como se juntássemos os N vetores – no caso exemplificado, os 3 vetores –, como na imagem a seguir:

0	1	2	3	.....	47	48	49
7.00	8.50	9.35	8.45	.....	9.00	7.90	9.60
9.70	9.20	8.95	9.00	.....	10.00	6.40	4.30
10.00	8.50	7.50	8.80	.....	9.20	8.00	9.00

Representação dos três vetores.

Além das 50 posições (em cada coluna), passamos a ter um segundo índice, que seriam 3 linhas – começando em 0 e indo até 2 no caso apresentado. Veja o resultado:

	0	1	2	3	.....	47	48	49
0	8.50	9.35	8.45	.....	.....	9.00	7.90	9.60
1	9.20	8.95	9.00	.....	.....	10.00	6.40	4.30
2	8.50	7.50	8.80	.....	.....	9.20	8.00	9.00

Representação dos três vetores.

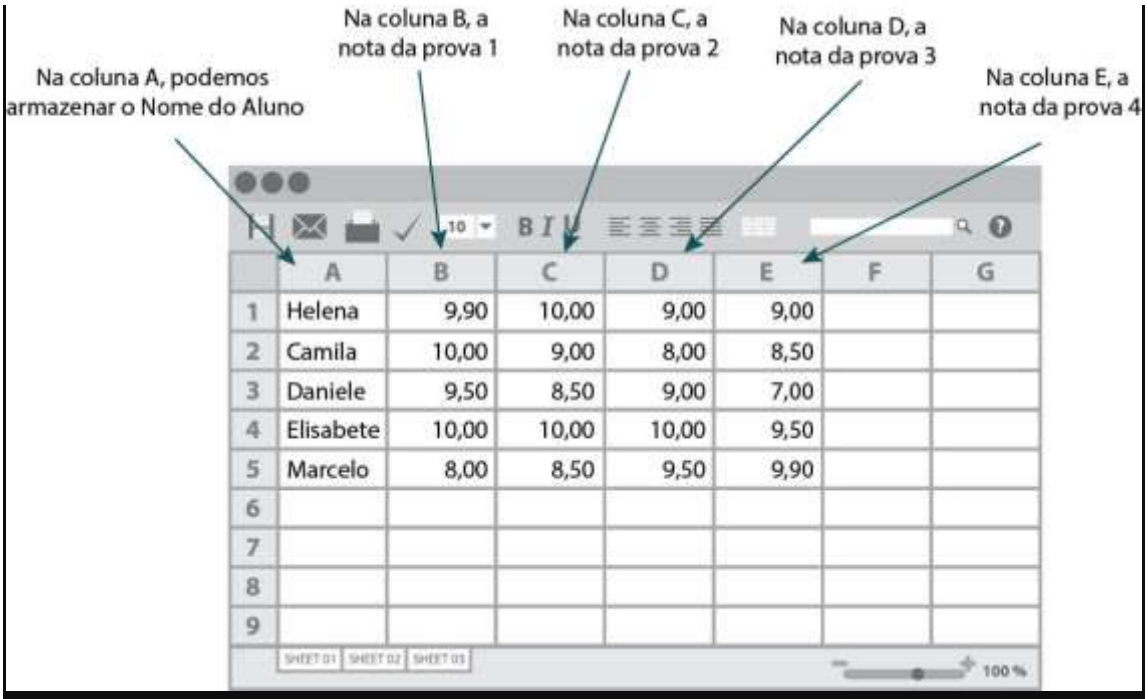
Conceito de matriz

A matriz é uma estrutura de dados homogênea, na qual usamos dois índices para acessar cada elemento.

Uma matriz é composta por **linhas e colunas**, tal qual uma planilha Excel, que também tem cada elemento referenciado por uma coluna (A, B, C, D, E) e uma linha (1 ... 10).

A diferença fundamental entre uma planilha Excel e uma matriz é que, na primeira, podemos armazenar dados de diferentes tipos.

Vamos ver um exemplo de planilha Excel com diversos tipos de dados, em que o nome do aluno usa um tipo de dado, e as notas das provas usam outro:



	A	B	C	D	E	F	G
1	Helena	9,90	10,00	9,00	9,00		
2	Camila	10,00	9,00	8,00	8,50		
3	Daniele	9,50	8,50	9,00	7,00		
4	Elisabete	10,00	10,00	10,00	9,50		
5	Marcelo	8,00	8,50	9,50	9,90		
6							
7							
8							
9							

Planilha Excel com diversos tipos de dado.

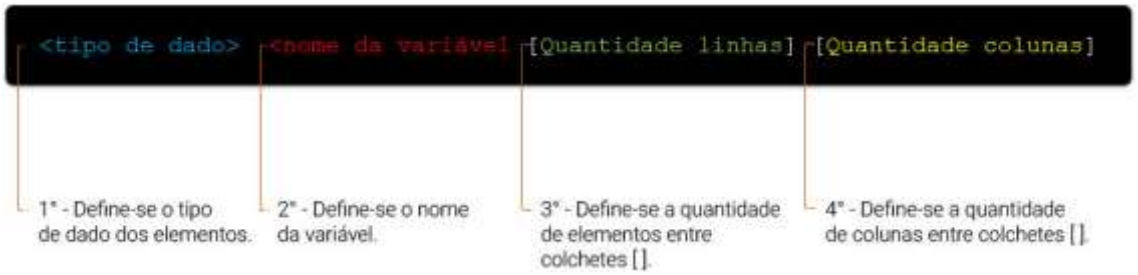
Agora um exemplo de planilha matriz com dados homogêneos, em que podemos armazenar somente dados de um único tipo:

	0	1	2	3
0	9,90	10,00	9,00	9,00
1	10,00	9,00	8,00	8,50
2	9,50	8,50	9,00	7,00
3	10,00	10,00	10,00	9,50
4	8,00	8,50	9,50	9,90

Planilha matriz com dados homogêneos.

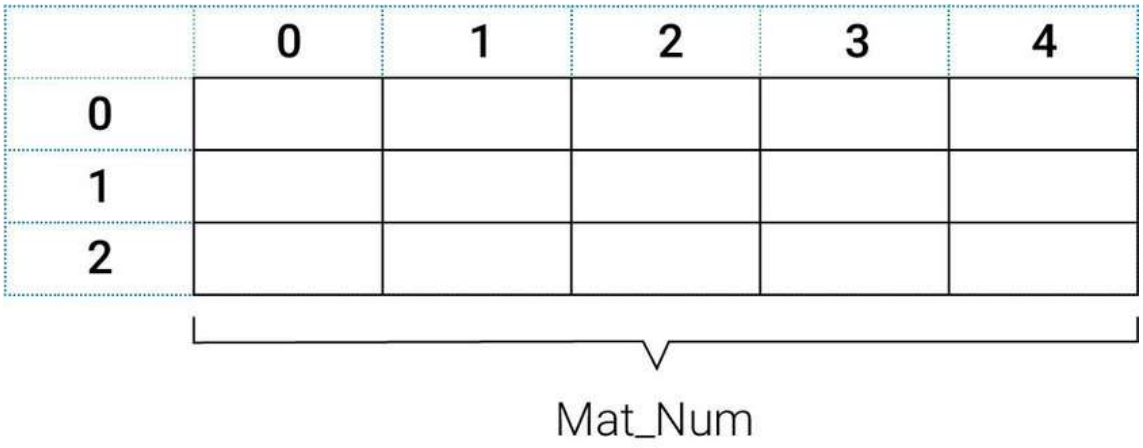
Declaração da matriz

Como qualquer variável, a **matriz precisa ser declarada** na maioria das linguagens de programação que assim o exigem, como é o caso da linguagem C. De forma geral, as matrizes são declaradas da seguinte maneira em Portugol Studio e na linguagem C:



Declarações em Portugol Studio e linguagem C das matrizes.

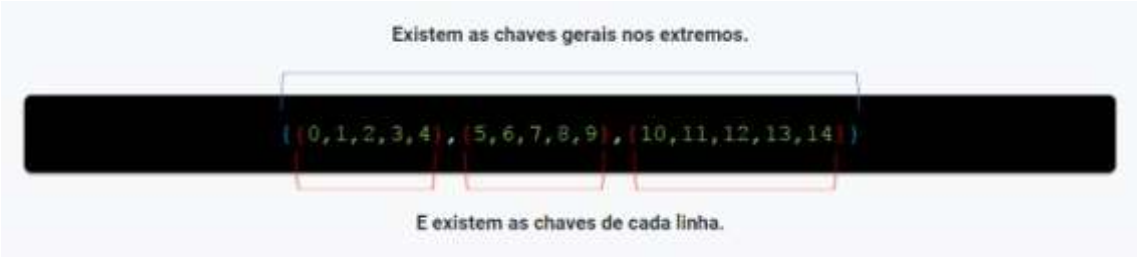
Vamos ver imagens que ilustram uma matriz de 3 linhas e 5 colunas de elementos inteiros, chamada Mat_Num:



Representação da Mat_Num.

Representação da Mat_Num.

Tanto no Portugol Studio quanto na linguagem C, o conjunto de elementos a serem armazenados na matriz deve estar definido entre chaves { }, como na seguinte imagem:



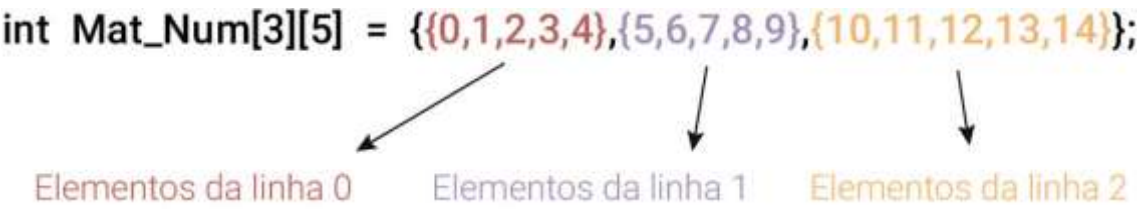
Definição entre chaves do conjunto de elementos.

A tabela a seguir mostra como ficará a matriz após a inicialização anterior de seus elementos:

	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	9
2	10	11	12	13	14

Tabela: Matriz preenchida após a inicialização.

A imagem a seguir mostra a inicialização da matriz anterior, com 3 linhas e 5 colunas:



Inicialização da matriz.

Vamos ver outros exemplos de declaração de matrizes nos seguintes códigos:

// Portugol Studio

inteiro números[100][200] → matriz de 100x200 elementos do tipo inteiro

real notas [20][30] → matriz de 20x30 elementos do tipo real

caractere vogais [5][10]→ matriz de 5x10 elementos do tipo caractere

// Linguagem C

int numeros[100][200];

float notas [20][30];

char vogais [5][10];

Assim como as variáveis de tipo simples e os vetores, as variáveis do tipo matriz também podem ser inicializadas durante sua declaração ou em um comando no trecho de código, conforme vemos a seguir:

// Portugol Studio

inteiro numeros[3][5] = {{10,12,14,16,18},{0,9,8,7,6,5},{11,12,13,14,15}}

ou

inteiro numeros[3][5]

numeros = {{10,12,14,16,18},{0,9,8,7,6,5},{11,12,13,14,15}}

// Linguagem C

int numeros[3][5] = {{10,12,14,16,18},{0,9,8,6,5},{11,12,13,14,15}};

ou

int numeros[3][5];

numeros = {{10,12,14,16,18},{0,9,8,6,5},{11,12,13,14,15}};

Acesso aos elementos da matriz

Os elementos de uma matriz são acessados, atribuídos, lidos ou exibidos, elemento a elemento, referenciando a linha e a coluna que ocupam, respectivamente.

Considere a matriz 3x5 (3 linhas x 5 colunas) de elementos do tipo inteiro, de nome mat, bem como os seguintes comandos, numerados de 1 a 15, tanto em Portugol Studio quanto em linguagem C:

/ / Portugol Studio

inteiro mat[3][5]

1 mat [0][0] = 1

2 mat [0][1] = 2

3 mat [0][2] = 3

4 mat [0][3] = 4

5 mat [0][4] = 5

6 mat [1][0] = 6

7 mat [1][1] = 7

8 mat [1][2] = 8

9 mat [1][3] = 9

10 mat [1][4] = 10

11 mat [2][0] = 11

12 mat [2][1] = 12

13 mat [2][2] = 13

14 mat [2][3] = 14

15 mat [3][3] = 15

// Linguagem C

mat[3][5];

1 mat [0][0] = 1;

2 mat [0][1] = 2;

3 mat [0][2] = 3;

4 mat [0][3] = 4;

5 mat [0][4] = 5;

6 mat [1][0] = 6;

7 mat [1][1] = 7;

8 mat [1][2] = 8;

9 mat [1][3] = 9;

10 mat [1][4] = 10;

11 mat [2][0] = 11;

12 mat [2][1] = 12;

13 mat [2][2] = 13;

14 mat [2][3] = 14;

15 mat [3][3] = 15;

Ao final da execução de cada comando associado às linhas numeradas de 1 a 15, teremos a matriz preenchida da seguinte forma:

	0	1	2	3	4
0	1	2	3	4	5
1	5	7	8	9	10
2	11	12	13	14	15

Tabela: Resultado da matriz preenchida.

Com base nesta tabela, acompanhe as considerações sobre matrizes:

- O 1º elemento ocupa a posição [0][0], linha 0 e coluna 0: é o número 1.
- O último elemento ocupa a posição [2][4], linha 2, coluna 4: é o número 15.

Considerando a matriz apresentada, veja o que vai acontecer com os comandos equivalentes em Portugol Studio e na linguagem C.

Comando para exibir um elemento da matriz

O comando exibe no dispositivo de saída o elemento que ocupa a posição de linha=2 e coluna=1. Avaliando a matriz, o elemento é 12 (doze), como no código a seguir:

// Portugol Studio

escreva (“Linha 2, coluna1 : “, mat[2][1])

// Linguagem C

printf(“%d Linha 2, coluna1:“, mat[2][1]);

Comando para atribuir valor a um elemento da matriz

O comando armazena o valor 90 no elemento da linha 2 e coluna 4. O elemento, anterior ao comando, é 15, que é substituído pelo 90, como no código a seguir

// Portugol Studio

mat[2][4]=90

// Linguagem C

Mat[2][4]=90;

A matriz ficará assim (observe o destaque em amarelo):

	0	1	2	3	4
0	1	2	3	4	5
1	5	7	8	9	10
2	11	12	13	14	90

Tabela: Representação da matriz.

Comando para ler um dado do dispositivo de entrada e armazenar na matriz

Há duas formas de ler um dado de dispositivo de entrada:

Leitura do dado de entrada direto para uma posição da matriz

O comando armazena o valor lido pelo dispositivo de entrada na linha 2, coluna 0, da matriz. O elemento anterior ao comando é 11, substituído pelo valor lido, como no código a seguir:

// Portugol Studio

```
leia (mat[2][0])
```

// Linguagem C

```
scanf ("%d",&mat[2][0]);
```

Se o usuário digitar o número 99, como ficará a matriz?

Observe o destaque em amarelo:

	0	1	2	3	4
0	1	2	3	4	5
1	5	7	8	9	10
2	99	12	13	14	90

Tabela: Representação da matriz.

Leitura do dado do dispositivo de entrada em uma variável

Nesse caso, atribui-se o conteúdo dessa variável a uma posição da matriz. O comando armazena em uma variável o valor lido pela digitação do usuário no dispositivo de entrada. Na sequência, armazena o conteúdo dessa variável na linha 1 e coluna 2 da matriz. O elemento anterior ao comando é 8, que é substituído pelo valor lido, como no código a seguir:

// Portugol Studio

```
leia (num)
mat[1][2]=num
```

// Linguagem C

```
scanf ("%d",&num);
```

mat[1][2]=num;

Se o usuário digitar o número 122, como ficará a matriz?

Observe o destaque em amarelo:

	0	1	2	3	4
0	1	2	3	4	5
1	5	7	122	9	10
2	99	12	13	14	90

Tabela: Representação da matriz.

Os comandos a seguir resultarão em erro, pois acessam posições (índices da matriz) que não são válidas:

// Portugol Studio

escreva (mat[0][5])

mat[3][4]=901

leia (mat[5][5])

// Linguagem C

printf("%d linha 0, coluna 5 :",mat[0][5]);

mat[3][4]=901;

scanf ("%d",&mat[5][5]);

Demonstração do uso de matrizes

Exemplo 1

O comando declara uma matriz para armazenar 2 notas de 30 alunos de uma turma, como nos códigos a seguir:

// Linguagem C

double mat[2][30];

ou

double mat[30][2];

Exemplo 2

O comando declara uma matriz para armazenar o sexo – masculino (M) ou feminino (F) – de 50 alunos de 3 turmas, como nos códigos a seguir

// Linguagem C

```
char mat[3][50];
```

ou

```
char mat[50][3];
```

Exemplo 3

O comando declara uma matriz para armazenar 5 apostas de 10 alunos de uma turma e inicializa cada aposta com 0 (zero), como nos códigos a seguir:

// Linguagem C

```
int mat[5][10] = {{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0}};
```

ou

```
float mat[10][5] =
{{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},
{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}};
```

Exemplo 4

Se fossem 3 notas de 100 alunos, em vez de 10, seria inviável inicializar a matriz com 3x100 zeros entre as chaves. Se fossem 500 alunos, mais inviável ainda.

Como inicializar cada elemento da matriz com 0 (zero)?

Nesses casos, podemos usar dois comandos de repetição: um para percorrer cada linha e, dentro desta, outro para percorrer cada coluna de uma linha e atribuir o valor 0 (zero) a cada uma.

O comando de repetição mais adequado, nesse caso, é o PARA (FOR), pois a repetição é para um número fixo e conhecido de vezes – no caso 3 linhas x 100 colunas. Veja:

```
for (linha=0;linha<3;linha++)
{
   for (coluna=0;coluna<100;coluna++)
   {
   mat[linha][coluna]=0;
```

```
    }
}
```

Exemplo 5

O trecho de código exibe no dispositivo de saída as 3 notas dos 100 alunos da turma, como nos códigos a seguir:

```
// Linguagem C
for (linha=0;linha<3;linha++) {
    printf("%d \n dados da linha= ",linha);<
    for (coluna=0;coluna<100;coluna++) {
        printf ("%d",mat[linha][coluna]);
    }
}
```

Exemplo 6

O trecho de código calcula e mostra a média de cada turma (soma das notas dos alunos dividida por 100, que é a quantidade de alunos), como nos códigos a seguir:

```
// Linguagem C
soma=0;
for (linha=0;linha<3;linha++) {
    for (coluna=0;coluna<100;coluna++) {
        soma=soma+mat[linha][coluna];
    }
    printf("%f media da turma = ",soma/100);
}
```

Vamos praticar

Vamos pôr em prática o que aprendemos no decorrer deste conteúdo. Sugerimos que você realize os seguintes procedimentos:

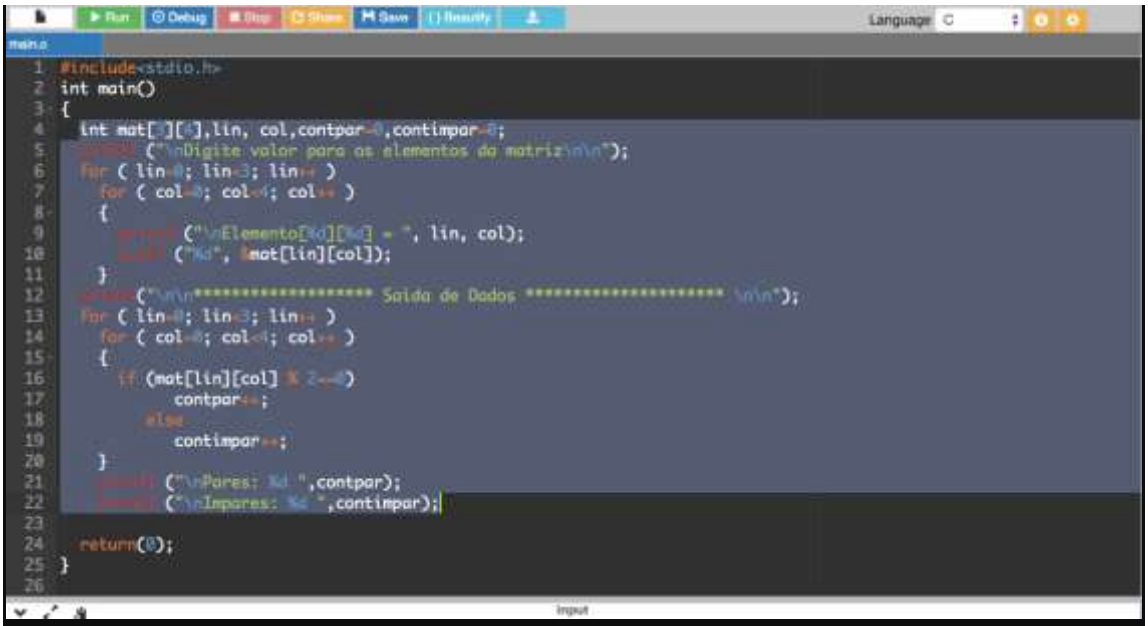
- Desenvolva o algoritmo em Portugol Studio e na linguagem C;
- Garanta que sua solução funcione, executando algumas vezes, com dados distintos;
- Veja a solução que sugerimos e compare com a sua.

Prática 1

Faça um algoritmo, no emulador a seguir, que leia dados inteiros e armazene-os em uma matriz 3x4. Em seguida, mostre a quantidade de números pares e ímpares armazenados na matriz.

```
#include <stdio.h>
int main() {
// Linguagem C
int mat[3][4],lin, col,contpar=0,contimpar=0;
printf ("\nDigite valor para os elementos da matriz\n\n");
for ( lin=0; lin<3; lin++ )
  for ( col=0; col<4; col++ )
  {
     printf ("\nElemento[%d][%d] = ", lin, col);
     scanf ("%d", &mat[lin][col]);
  }
printf("\n\n********* Saida de Dados ********* \n\n");
for ( lin=0; lin<3; lin++ )
  for ( col=0; col<4; col++ )
  {
     if (mat[lin][col] % 2==0)
        contpar++;
     else
        contimpar++;
  }
printf ("\nPares: %d ",contpar);
printf ("\nImpares: %d ",contimpar);
   return 0;
}
```

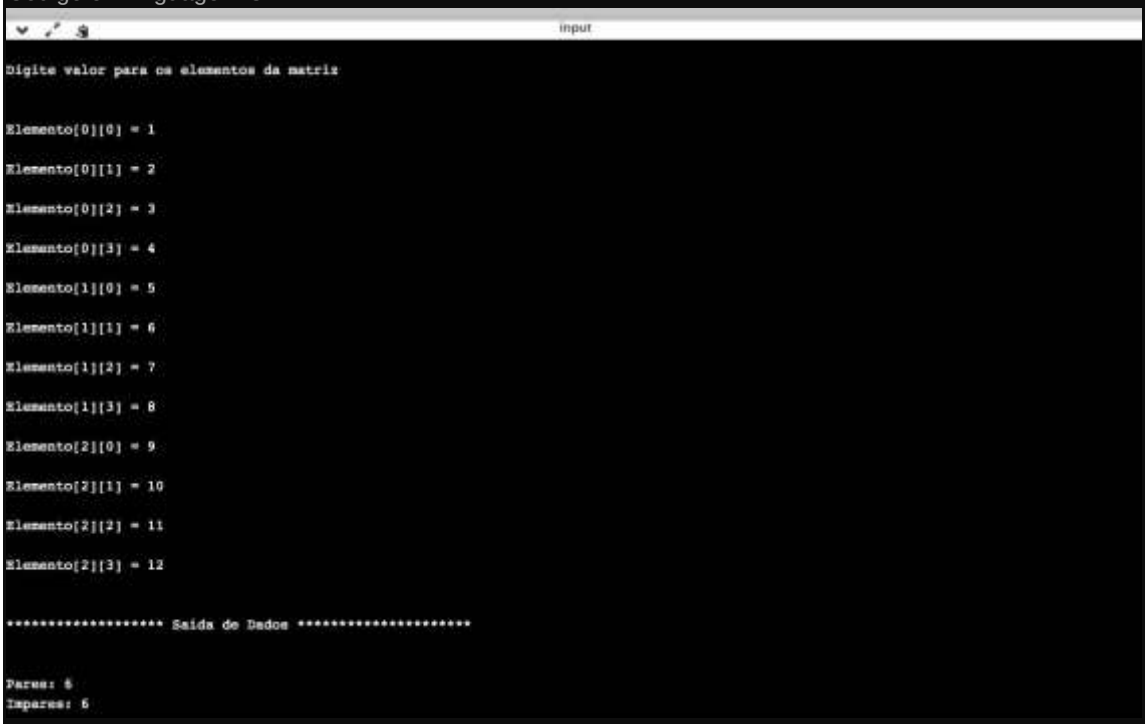
Veja as telas com o código em linguagem C, na ferramenta online, e a execução do programa:



```c
#include<stdio.h>
int main()
{
  int mat[ ][ ],lin, col,contpar=0,contimpar=0;
  printf ("\nDigite valor para os elementos da matriz\n\n");
  for ( lin=0; lin<3; lin++ )
    for ( col=0; col<4; col++ )
    {
      printf ("\nElemento[%d][%d] = ", lin, col);
      scanf ("%d", &mat[lin][col]);
    }
  printf("\n\n****************** Saida de Dados ******************* \n\n");
  for ( lin=0; lin<3; lin++ )
    for ( col=0; col<4; col++ )
    {
      if (mat[lin][col] % 2==0)
          contpar++;
        else
          contimpar++;
    }
    printf ("\nPares: %d ",contpar);
    printf ("\nImpares: %d ",contimpar);

  return(0);
}
```

Código em linguagem C.



```
Digite valor para os elementos da matriz

Elemento[0][0] = 1

Elemento[0][1] = 2

Elemento[0][2] = 3

Elemento[0][3] = 4

Elemento[1][0] = 5

Elemento[1][1] = 6

Elemento[1][2] = 7

Elemento[1][3] = 8

Elemento[2][0] = 9

Elemento[2][1] = 10

Elemento[2][2] = 11

Elemento[2][3] = 12


****************** Saida de Dados *******************


Pares: 6
Impares: 6
```

Execução do programa.

Prática 2

Faça, no emulador a seguir, um algoritmo que leia números inteiros e armazene-os na matriz 4x4. Porém, na diagonal principal, não armazene o número lido, e sim um 0 (zero).

Na diagonal principal, os elementos têm linha = colona: [0][0], [1][1], [2][2], [3][3].

```
#include <stdio.h>
// video 1 do modulo 2

int main() {

   int mat[4][4],lin,col;

   printf ("\nDigite valor para os elementos da matriz\n");

   for (lin=0;lin<4;lin++)

      for (col=0;col<4;col++)

      if (lin==col) {
         printf ("Elemento[%d][%d] = 0 \n",lin,col);
         mat[lin][col]=0;
      } else {
         printf ("Elemento[%d][%d] = ",lin,col);
         scanf ("%d",&mat[lin][col]) ;
      }

   printf ("\nListagem dos elementos da matriz\n");

   for (lin=0;lin<4;lin++)
      for (col=0;col<4;col++)
      printf("\nElemento[%d][%d] = %d",lin,col,mat[lin][col]);
}
```

Prática 3

Faça um algoritmo que leia uma matriz 4x4 de números inteiros. Gere uma segunda matriz, na qual as linhas são as colunas da matriz 1, e as colunas são as linhas da matriz 1.

Matriz 1				Matriz 2 gerada			
1	2	3	4	1	5	9	13
5	6	7	8	2	6	10	14
9	10	11	12	3	7	11	15
13	14	15	16	4	8	12	16

Matriz 1 e 2 geradas.

```
#include <stdio.h>

// video 2 do modulo 2

int main() {

   int mato[4][4],matg[4][4],lin,col;
   printf ("\n Digite a matriz original \n");

   for (lin=0;lin<4;lin++)

      for (col=0;col<4;col++) {
         scanf ("%d",&mato[lin][col]);
         matg[col][lin]=mato[lin][col];
      }

   printf ("\n Matriz gerada \n");

   for (lin=0;lin<4;lin++) {
      for (col=0;col<4;col++)
      printf ("%d ",matg[lin][col]);
      printf ("\n");
   }
```

```
}
```

Prática 4

Faça, no emulador a seguir, um algoritmo que leia dados e armazene em uma matriz 3x3 de números inteiros. Em seguida, mostre os elementos que sejam iguais ao maior número armazenado na matriz.

1° forma:

```
#include <stdio.h>


// Video 3 do modulo 2 (solucao 2)


int main() {


   int mat1[3][3],lin, col,maior=0,contigual=0;
   printf("\n Digite valor para os elementos da matriz \n\n ");


   for ( lin=0; lin<3; lin++ )
      for ( col=0; col<3; col++ ) {
         printf ("Elemento[%d][%d] = ",lin,col);
         scanf ("%d", &mat1[lin][col]);
         if (mat1[lin][col]>maior)
            maior=mat1[lin][col];
         }
   for (lin=0; lin<3; lin++ )
      for (col=0; col<3; col++ ) {
         if (mat1[lin][col]==maior)
         contigual++;
      }
   printf("\n Maior: %d ",maior);
   printf("\n = maior: %d ",contigual);
}
```

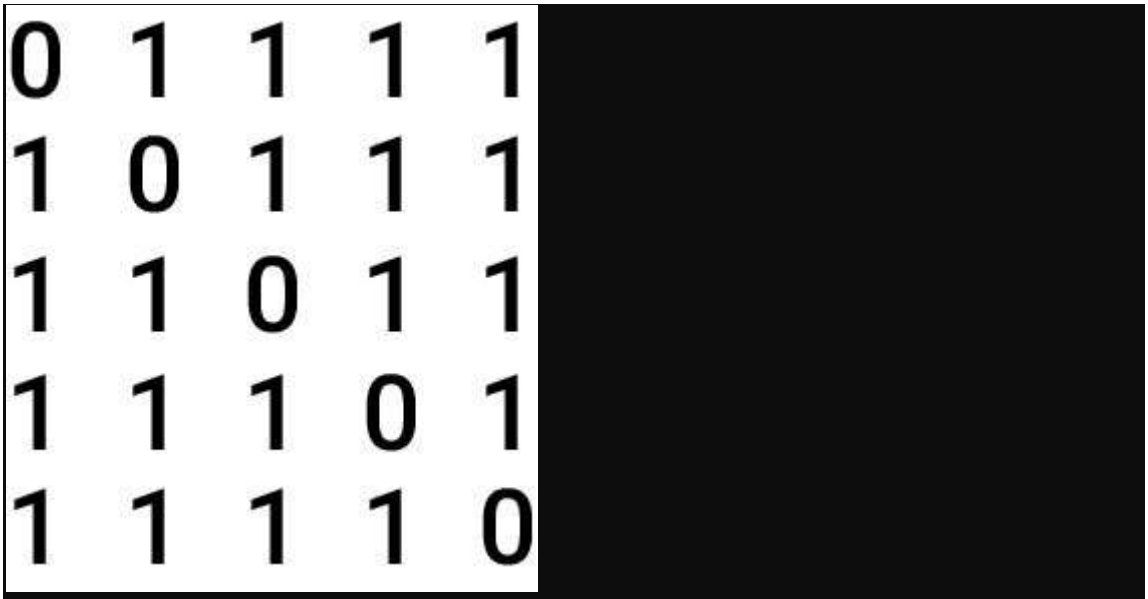
2° forma:

```
#include <stdio.h>
// video 3 do modulo 2
int main()
{
   int mat[3][3],lin, col,maior=0,contigual=0;
   printf ("\nDigite valor para os elementos da matriz\n\n");
   for ( lin=0; lin<3; lin++ )
   for ( col=0; col<3; col++ )
   {
      printf ("Elemento[%d][%d] = ", lin, col);
      scanf ("%d", &mat[lin][col]);
      if (mat[lin][col]>maior)
      {
         maior=mat[lin][col];
         contigual=1;
      }
      else
         if (mat[lin][col]==maior)
         contigual++;
      }

   printf("\n Maior: %d ",maior);
   printf("\n Ocorrenias do maior: %d ",contigual);
      return 0;
}
```

Prática 5

Faça um programa que gere uma matriz 5x5, conforme esta sequência:



Matrix 5x5.

A seguir, o emulador para execução:

```
#include <stdio.h>
// video 4 do modulo 2
int main()
{
   int mat[5][5],lin, col;
   for (lin=0; lin<5; lin++ )
   {
      for ( col=0; col<5; col++)
      {
         mat[lin][col]=1;
         if (lin==col)
            mat[lin][col]=0;
         printf("%d",mat[lin][col]);
      }
      printf ("\n");
   }
   return 0;
}
```