Ponteiros e Matrizes
Algoritmos e Programação II

Ponteiros para elementos de uma matriz

- Uma matriz com ***n*** linhas em linguagem C é armazenada como um vetor na memória, como na figura a seguir.
- Essa representação nos ajuda a trabalhar com ponteiros e matrizes.

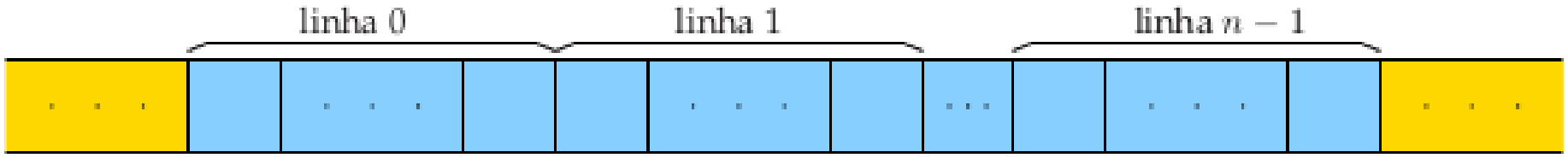


Figura 13.1: Representação da alocação de espaço na memória para uma matriz.

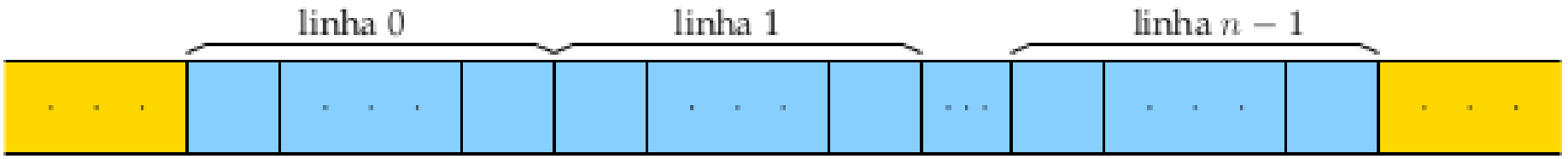


Figura 13.1: Representação da alocação de espaço na memória para uma matriz.

```
#define LINHAS 3
#define COLUNAS 3

int   A[LINHAS][COLUNAS];
```

Vamos considerar esses valores para todos os slides à frente

	0	1	2
0			
1			
2			

	0	1	2
0	00	01	02
1	10	11	12
2	20	21	22

	0	1	2
0	201	202	203
1	204	205	206
2	207	208	209

$A[\ i\ ][\ j\ ]$
$A[0][2]$
$A[1][1]$

Vamos supor em nossos exemplos:
inteiro → 1byte

Ponteiros para elementos de uma matriz

- Suponha que queremos inicializar uma matriz de inteiros com 0.

int A[LINHAS][COLUNAS];

```
int i, j;
.
.
.
for (i = 0; i < LINHAS; i++)
   for (j = 0; j < COLUNAS; j++)
      A[i][j] = 0;
```

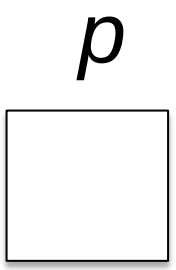
Ponteiros para elementos de uma matriz

- Mas, se vemos a matriz A da forma como é armazenada na memória, isto é, como um **vetor unidimensional**, podemos trocar o par de estruturas de repetição por uma única estrutura de repetição:

```
int *p;
:
for (p = &A[0][0]; p <= &A[LINHAS-1][COLUNAS-1]; p++)
    *p = 0;
```

	0	1	2
0	201	202	203
1	204	205	206
2	207	208	209

	0	1	2
0			
1			
2			



Processamento das linhas de uma matriz

- Podemos **COM PONTEIRO** processar uma linha da matriz – ou seja, percorrê-la, visitar os conteúdos dos compartimentos, usar seus valores, modificá-los, etc. Por exemplo, para visitar os elementos da linha ***i*** de uma matriz A podemos usar um ponteiro ***p*** apontar para o elemento da linha ***i*** e da coluna **0** da matriz A:

$$p = \&A[i][0];$$

Ou simplesmente

$$p = A[i]$$

$A[i]$ é um ponteiro para o primeiro elemento de cada linha ***i***.

Processamento das linhas de uma matriz

3 3

```
int A[LINHAS][COLUNAS], *p, i;
⋮ i = 1;
for (p = A[i]; p < A[i] + COLUNAS; p++)
    *p = 0;
```

	0	1	2
0	201	202	203
1	204	205	206
2	207	208	209

p = 204

Processamento das linhas de uma matriz

- Como $A[i]$ é um ponteiro para a linha ***i*** da matriz A, pode-mos passar $A[i]$ para uma função que espera receber um **vetor como argumento**. OU SEJA, uma função que foi projetada para trabalhar com um vetor também pode trabalhar com **uma linha de uma matriz**.

- Dessa forma, a função ***max*** (**calcula o maior valor de um vetor**) pode ser chamada com a linha i da matriz A como argumento:

M = ***max***(COLUNAS, $A[i]$);

```
int max(int n, int v[MAX])
{
    int i, maior;
    maior = v[0];
    for (i = 1; i < n; i++)
        if (v[i] > maior)
            maior = v[i];
    return maior;
}
```

Processamento das linhas de uma matriz

- Como $A[i]$ é um ponteiro para a linha ***i*** da matriz A, podemos passar $A[i]$ para uma função que espera receber um **vetor como argumento**. OU SEJA, uma função que foi projetada para trabalhar com um vetor também pode trabalhar com **uma linha de uma matriz**.

- Dessa forma, a função ***max*** (**calcula o maior valor de um vetor**) pode ser chamada com a linha i da matriz A como argumento:

M = ***max***(COLUNAS, $A[i]$);

Se $i=1$

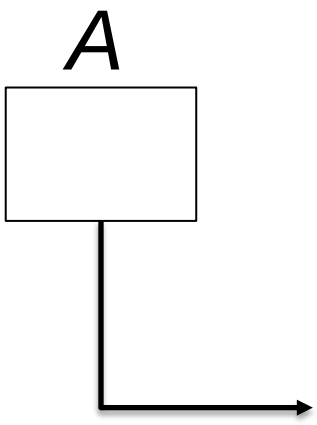
M = ***max***(COLUNAS, $A[1]$);

	0	1	2
0	201	202	203
1	204	205	206
2	207	208	209

Identificadores de matrizes como ponteiros

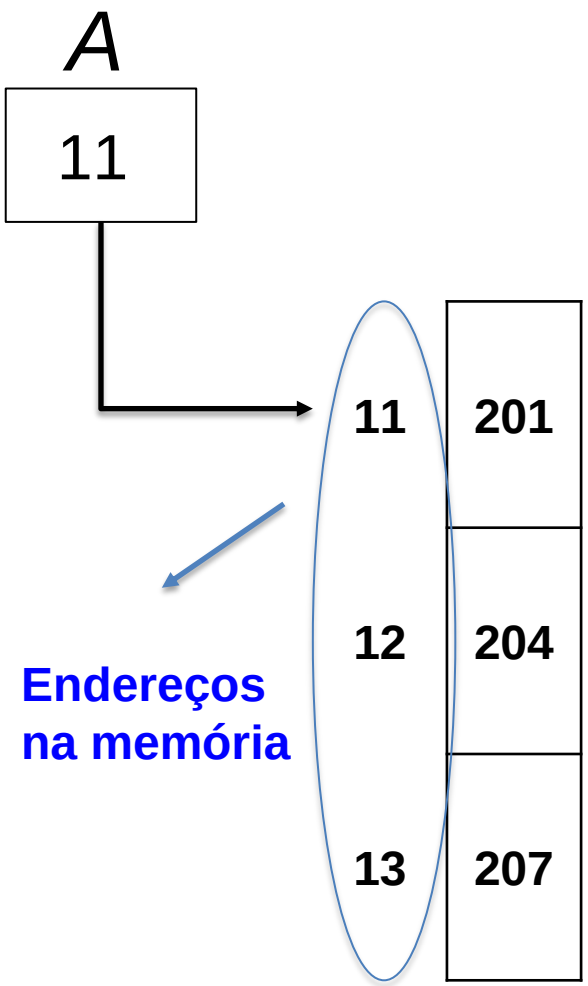
- Neste caso, *A* NÃO é um ponteiro para *A* **[0][0];**
- é um ponteiro para *A* **[0]** .
- Isso faz mais sentido se olharmos sob o ponto de vista da linguagem C, que considera *A* não como uma matriz bidimensional, mas como um vetor.

int *A*[LINHAS][COLUNAS];



0	201
1	204
2	207

201	202	203
204	205	206
207	208	209



201	202	203
204	205	206
207	208	209

```
int *p;
p = A;
```

```
int **p;
p = A;
```

```
int *p;
p = *A;
```

EXERCÍCIOS

```
#define MAX 5


int main()
{
      int matriz[MAX][MAX], n;


      scanf(“%d”, &n);

}
```

201	**202**	**203**	**204**	**205**
206	**207**	**208**	**209**	**210**
211	**212**	**213**	**214**	**215**