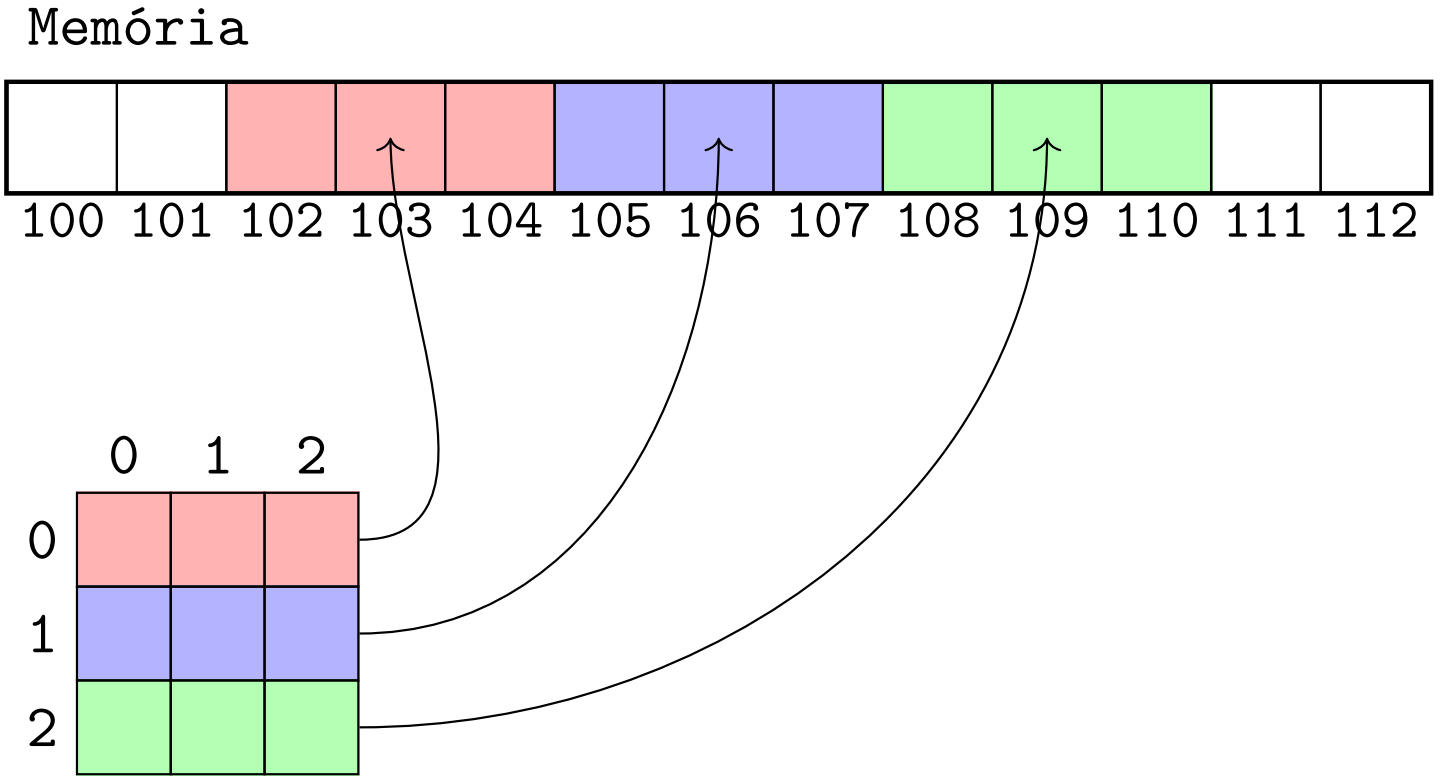
Fundamentos de Programação
Matrizes e funções

Dainf - UTFPR

Profa. Leyza B. Dorini
Prof. Bogdan T. Nassu

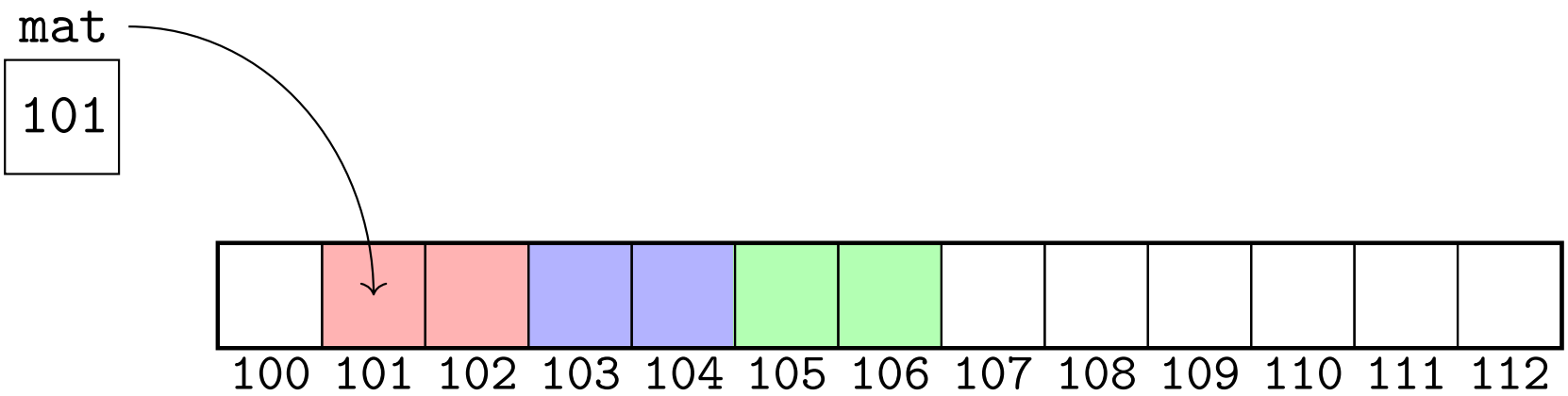
Matrizes

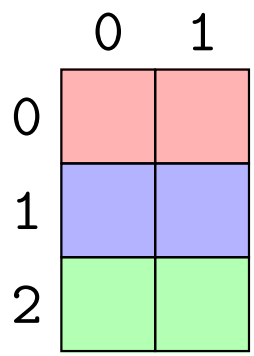
Como vimos, ao declarar uma matriz de tamanho estático (ou seja, em que a dimensão não é definida em tempo de execução), ela vai estar localizada na memória da mesma forma que um vetor: **em posições contíguas**. Veja a ilustração:



Matrizes

Ao declarar uma matriz `int mat[3][2]`, por exemplo, serão alocadas 6 posições contíguas de memória. A variável `mat` é um ponteiro que armazena o endereço da primeira posição de memória alocada.





Matrizes e funções

Desta forma, sem precisarmos usar uma notação especial, matrizes são sempre passadas por **referência**.

Atenção

Portanto, ao passar matrizes como parâmetro para funções seu conteúdo pode ser alterado dentro da função (pois estamos passando, na realidade, o endereço de início do espaço alocado para os elementos da matriz).

Exemplo

```
1
2  int main(){
3      int m[2][3] = {{1, 2, 3}, {3, 6, 7}};
4      imprimeMatriz( m,2,3);
5      mudaMatriz(m,2,3);
6      imprimeMatriz(m,2,3);
7
8      return 0;
9  }
```

Atenção:
ao passar uma matriz por parâmetro,
devemos colocar apenas o seu identificador
(e não &m ou m[][]).

Recebendo matrizes por parâmetro em funções

Qual a sintaxe para a matriz no protótipo da função?

Ao passar um vetor como parâmetro, vimos que não é necessário fornecer o seu tamanho na declaração da função. Para matrizes, a possibilidade de não informar o tamanho na declaração **se restringe à primeira dimensão apenas**. Você consegue justificar o porquê?

Para a função imprimeMatriz()

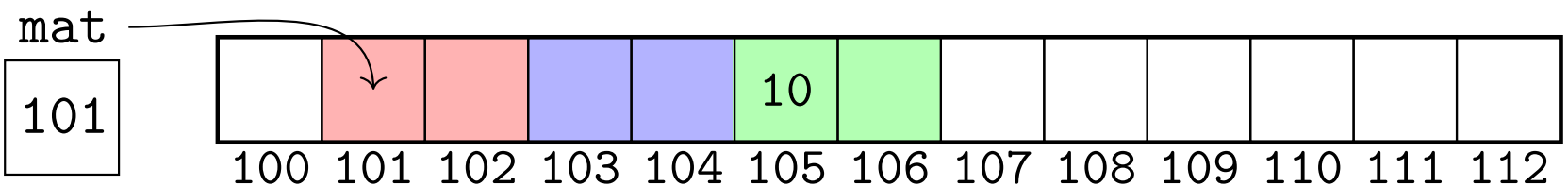
```
void imprimeMatriz(int mat[][3], int nl, int nc) {
  //VALIDO!!
}
void imprimeMatriz(int mat[2][3], int nl, int nc) {
  //VALIDO!!
}
void imprimeMatriz(int mat[][], int nl, int nc) {
  //INVALIDO!!
}
```

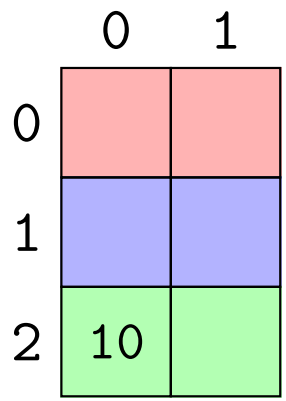
Linearização de índices

Dado que, para acessar um elemento no índice `mat[i][j]`, é realizado o mapeamento para o endereço

```
mat + i * NCOL + j,
```

a função **precisa** saber a quantidade de colunas!





Exemplo Completo

```
1  void mudaMatriz(int m[][3], int nl, int nc){
2      int i, j;
3      for(i=0; i<nl; i++)
4          for(j=0; j<nc; j++)
5              m[i][j]=m[i][j]*2;
6  }
7
8
9  void imprimeMatriz(int m[][3], int nl, int nc){
10     int i, j;
11     for(i=0; i<nl; i++){
12         for(j=0; j<nc; j++)
13                 printf("%3d ", m[i][j]);
14         printf("\n");
15     }
16 }
17
18
```

Exemplo Completo

```
1  int main(){
2      int m[2][3] = {{1, 2, 3}, {3, 6, 7}};
3      imprimeMatriz(m,2,3);
4      mudaMatriz(m,2,3);
5      imprimeMatriz(m,2,3);
6
7      return 0;
8  }
```

Saída:

```
  1   2   3
  3   6   7

  2   4   6
  6  12  14
```

Ao passar matrizes por parâmetro, a função precisa saber a sua dimensão (linhas e colunas)

Matrizes com tamanho definido por uma constante

Nos exemplos anteriores, observe que **passamos como parâmetro, além da matriz, a quantidade de linhas e de colunas!** Se tais valores fossem definidos por constantes, por exemplo, esses argumentos não seriam necessários! Exemplo para a função `mudaMatriz()`.

```
1  #define NL 2
2  #define NC 3
3
4  void mudaMatriz(int m[][NC]){
5      int i, j;
6      for(i=0; i<NL; i++)
7          for(j=0; j<NC; j++)
8              m[i][j]=m[i][j]*2;
9  }
```

Atenção: note que podemos explorar o uso da constante aqui (assim, caso o valor seja alterado, o programa vai continuar consistente)

Matrizes com tamanho definido por uma constante

Contudo, é uma boa prática passar a dimensão como parâmetro de entrada. Dessa forma, a função fica mais genérica (no sentido de permitir que a operação sendo executada pela função seja realizada em apenas parte da matriz).

```
1  #define NL 2
2  #define NC 3
3  void imprimeMatriz(int m[][NC], int nl, int nc){
4      int i, j;
5      for(i=0; i<nl; i++){
6          for(j=0; j<nc; j++)
7                  printf("%3d ", m[i][j]);
8          printf("\n");
9      }
10 }
11
12 int main(){
13     int m[NL][NC] = {{1, 2, 3}, {3, 6, 7}};
14     imprimeMatriz(m,1,2); //apenas parte da matriz é impressa
15     return 0;
16 }
```

Como retornar uma matriz?

Como retornar um matriz por parâmetro?

Até aprendermos alocação dinâmica, passe a matriz "resposta" por parâmetro.

Suponha, por exemplo, uma função que armazena em uma nova matriz (`output`) o dobro dos elementos da matriz de entrada (`input`).

```
void testa(int input[][NC], int output[][NC], int nl, int nc)
{
    int i, j;
    for(i=0; i<nl; i++)
       for(j=0; j<nc; j++)
          output[i][j] = input[i][j]*2;
}
```

Note que, ao invés de criarmos a matriz `output` dentro da função e retorná-la, ela é recebido como parâmetro de entrada (ou seja, é declarada na própria função que invocou `testa()`).

Exemplos

Exemplo: retorno do maior valor em uma matriz (p1)

```
1  #define NL 3
2  #define NC 5
3
4  int maxValue(int m[][NC], int nl, int nc) {
5
6      int i, j,
7          max;
8
9      max = m[0][0]; //inicializa maior como 1o elemento
10     for (i = 0; i < nl; i++) {
11         for(j = 0; j < nc; j++)
12            if (m[i][j] > max)
13                max = m[i][j];
14     }
15
16     return max;
17 }
```

Exemplo: retorno do maior valor em uma matriz (p2)

```
1  int main() {
2      int m[NL][NC] = ... // suponha uma inicializacao,
3          max;
4
5      max = maiorValor(m, NL, NC);
6
7      printf("Maior valor: %d\n", max);
8      return 0;
9  }
```

E agora..

... fica como tarefa fazer a lista de exercícios!