

# Dominando Estruturas de Dados 1

## Matrizes Estáticas e Dinâmicas

Prof. Samuel Martins (Samuka)  
@xavecoding @hisamuka

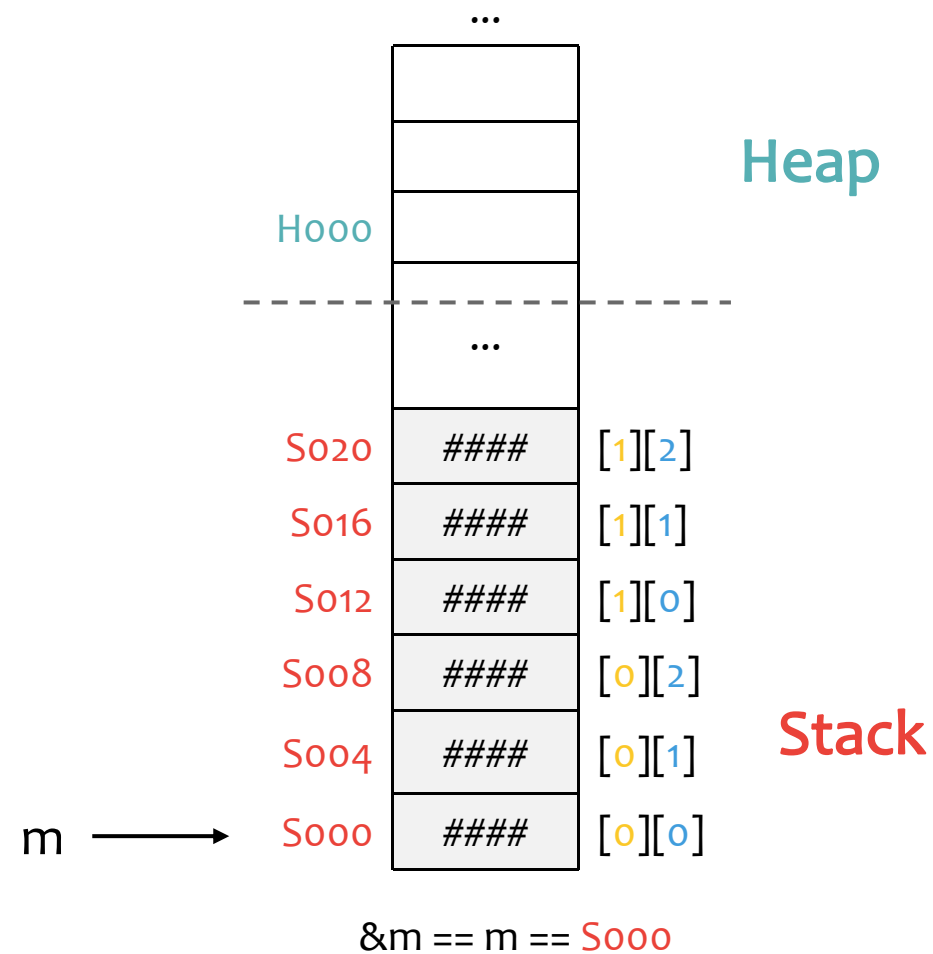


# Alocação Estática

```
int m[2][3];
```

|     |   | 0 | 1 | 2 |
|-----|---|---|---|---|
| m = | 0 | # | # | # |
|     | 1 | # | # | # |

# = “lixo” de memória



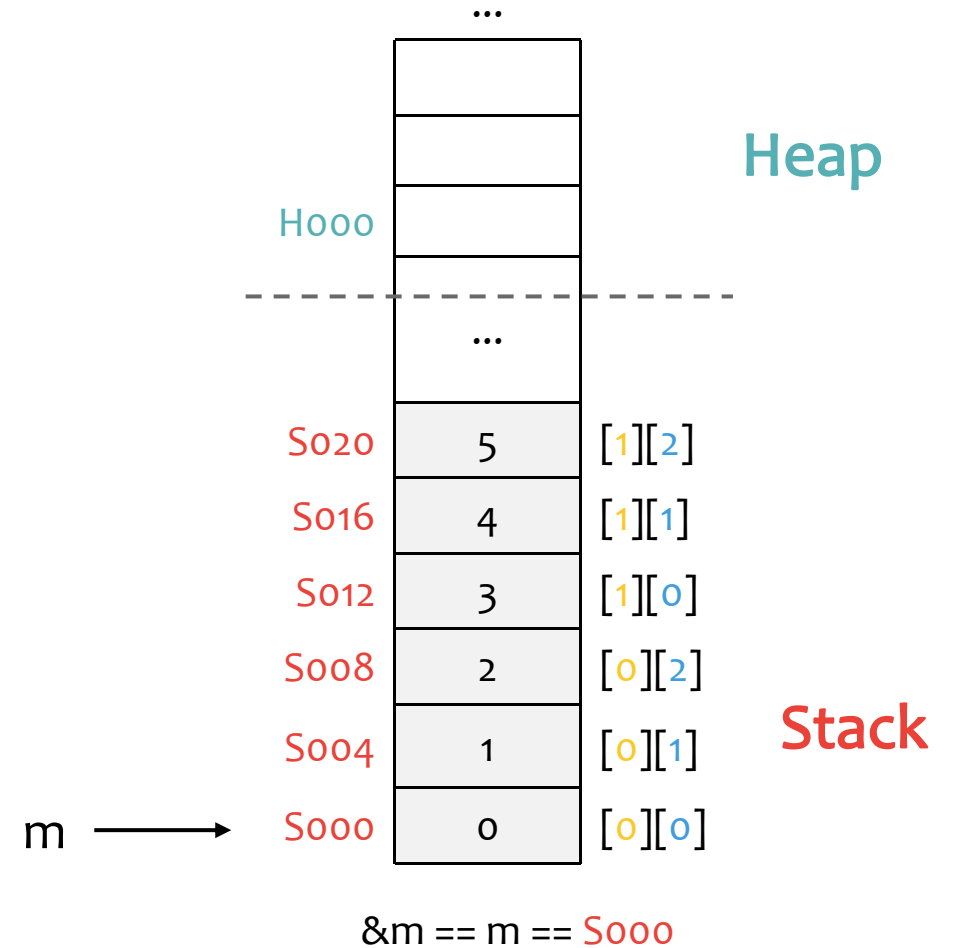
# Alocação Estática

```
int m[2][3] = {{0, 1, 2}, {3, 4, 5}};
```

m =

|   | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 1 | 2 |
| 1 | 3 | 4 | 5 |

**Obs:** Só é possível instanciar a matriz na **alocação estática**



# Let's code!

Codifique um programa que aloque uma **matriz estática** e imprima o endereço de cada elemento da matriz.

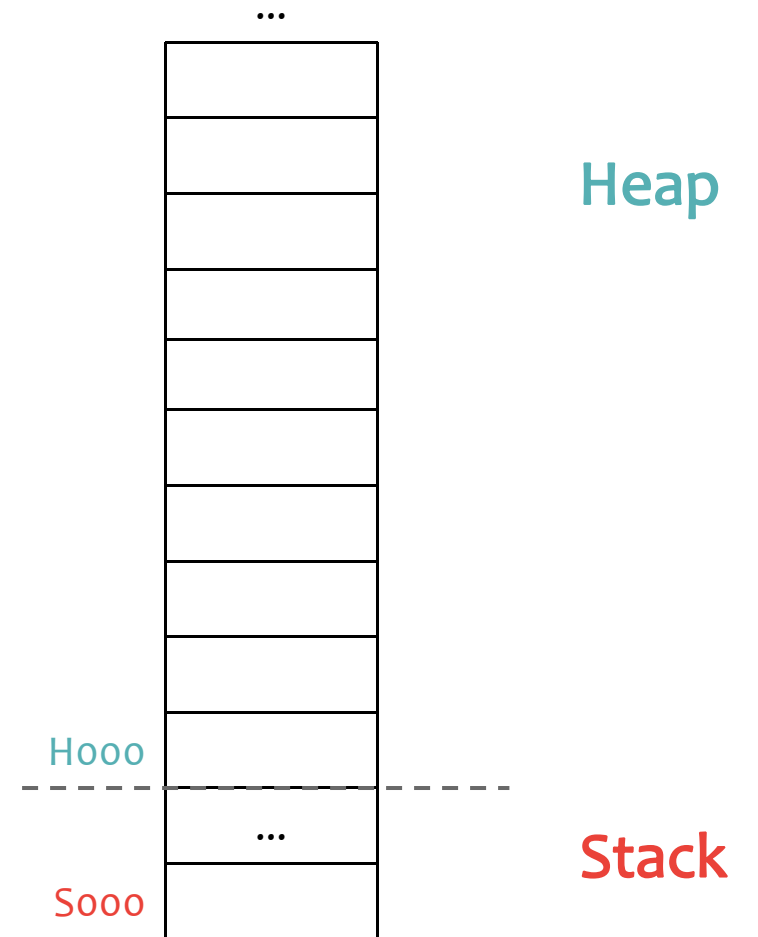
# Alocação Dinâmica

- Responsável pela **alocação** de uma matriz na memória **Heap**;
- Queremos um *Vetor de Vetores / Array de Arrays*

# Alocação Dinâmica

```
int **m = NULL; // nrow = 2; // ncol = 3

m = (int**) calloc(nrow, sizeof(int*));
for (int i = 0; i < nrow; i++) {
    m[i] = (int*) calloc(ncol, sizeof(int));
}
```



# Alocação Dinâmica

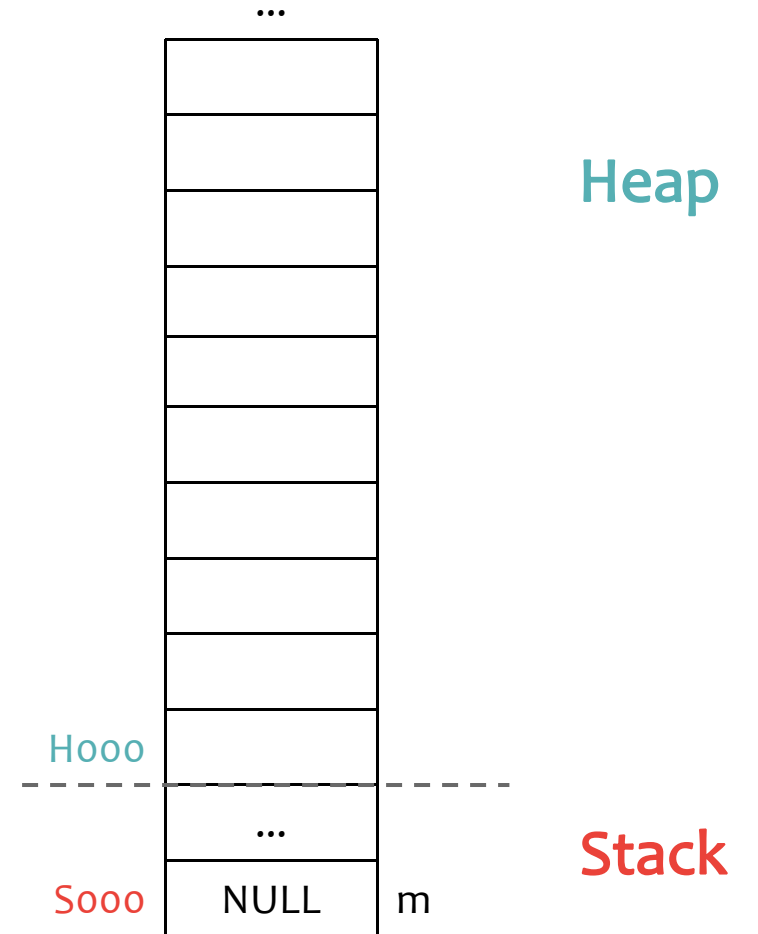
```
→ int **m = NULL; // n rows = 2; // n cols = 3

m = (int**) calloc(nrows, sizeof(int*));
for (int i = 0; i < nrows; i++) {
    m[i] = (int*) calloc(ncols, sizeof(int));
}
```

m

## Boa Prática de Programação

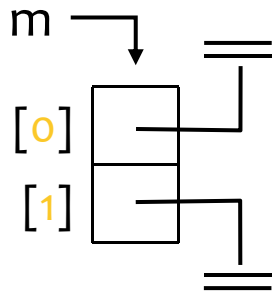
- Inicializar os ponteiros para **NULL**, para não apontar para “lixo”;
- **NULL** é o valor 0 para ponteiros.



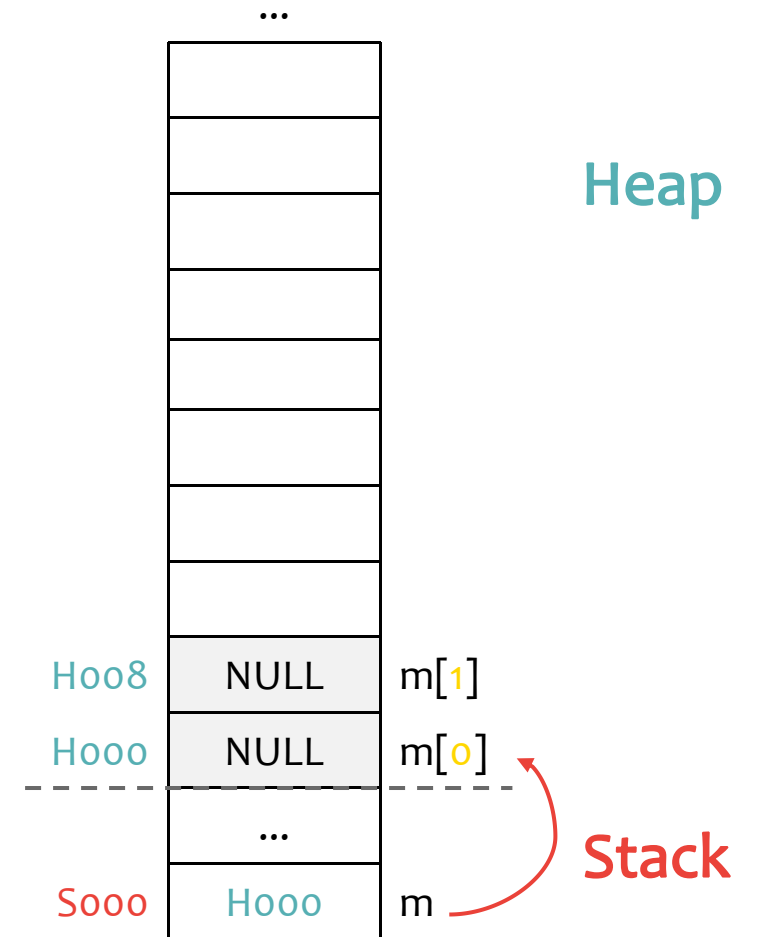
# Alocação Dinâmica

```
int **m = NULL; // nrows = 2; // ncols = 3

m = (int**) calloc(nrows, sizeof(int*));
for (int i = 0; i < nrows; i++) {
    m[i] = (int*) calloc(ncols, sizeof(int));
}
```



**calloc** atribui **NULL** (valor 0)  
para os ponteiros

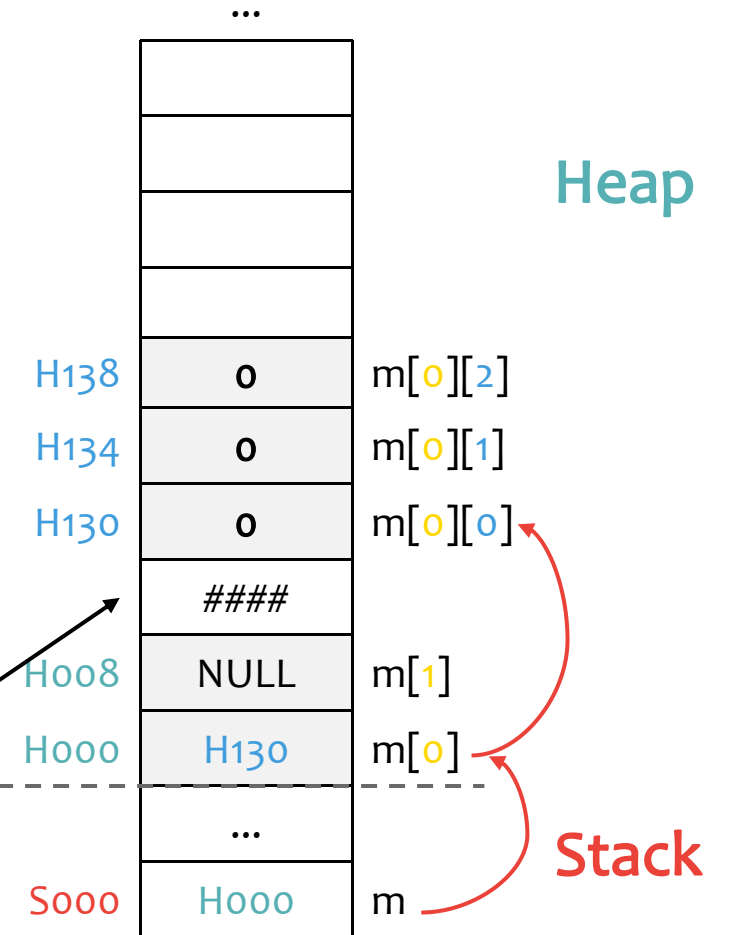
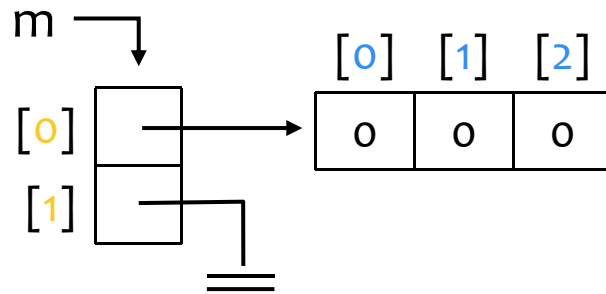




# Alocação Dinâmica

```
int **m = NULL; // nrow = 2; // ncol = 3

m = (int**) calloc(nrow, sizeof(int*));
for (int i = 0; i < nrow; i++) {
    m[i] = (int*) calloc(ncol, sizeof(int));
}
```

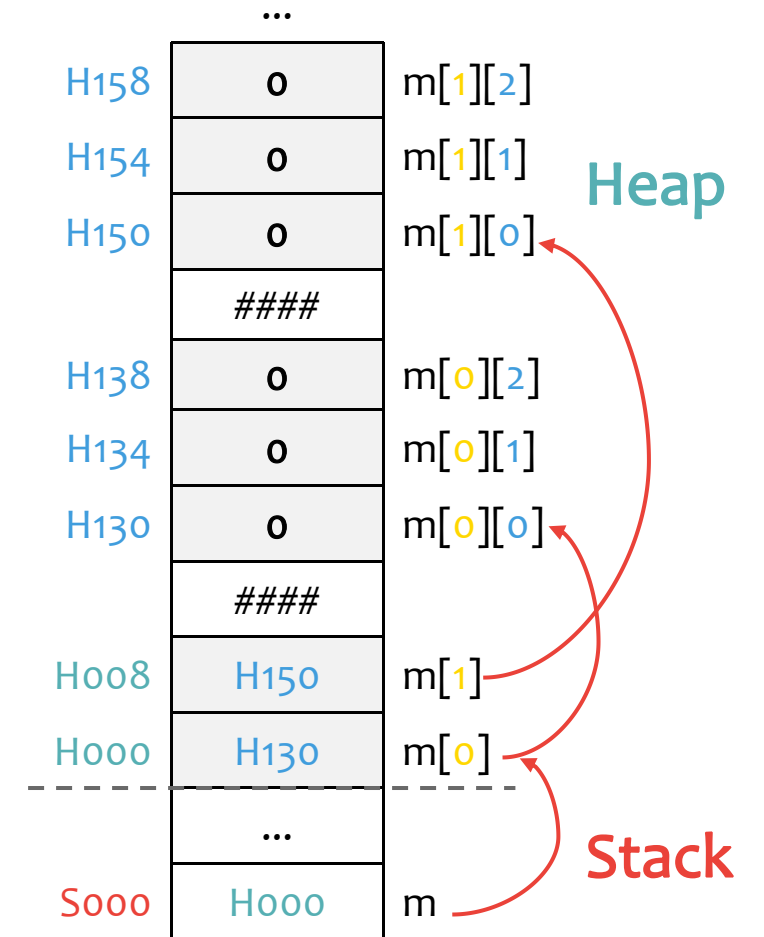
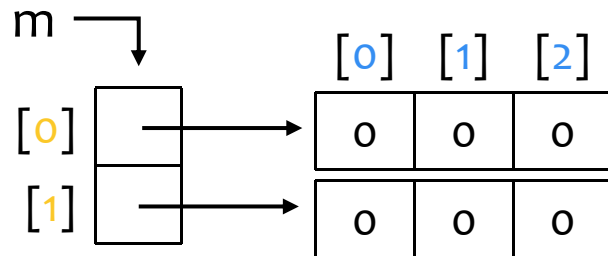


Não há garantias que o espaço alocado será **contíguo**

# Alocação Dinâmica

```
int **m = NULL; // nrow = 2; // ncol = 3

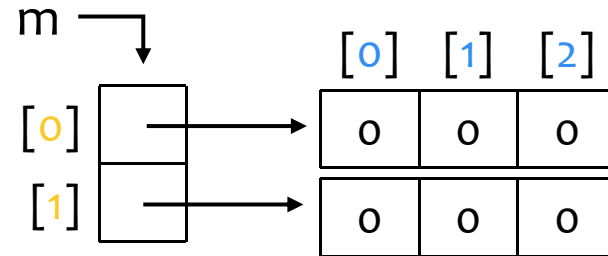
m = (int**) calloc(nrow, sizeof(int*));
for (int i = 0; i < nrow; i++) {
    m[i] = (int*) calloc(ncol, sizeof(int));
}
```



Você pode acessar normalmente o valor da matriz: `m[i][j]`

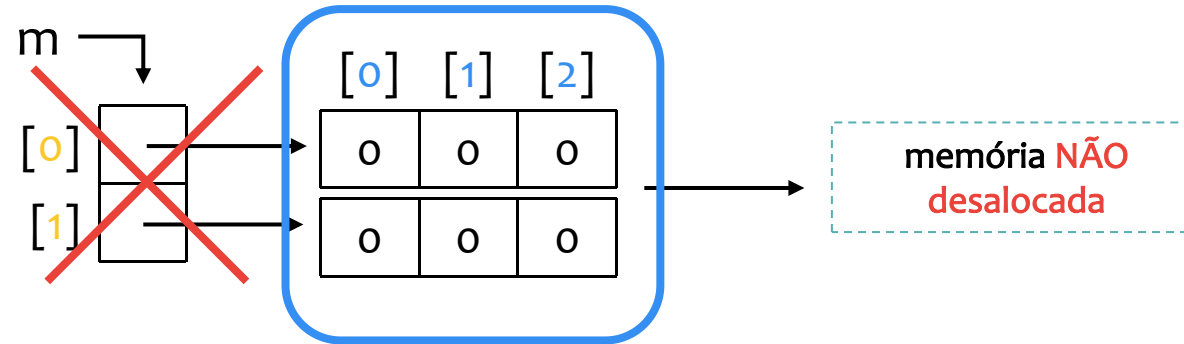
# Desalocando uma Matriz Dinâmica

- O que acontece se apenas fizemos: `free(m)`?



# Desalocando uma Matriz Dinâmica

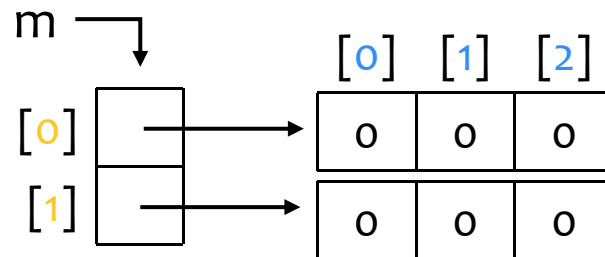
- O que acontece se apenas fizemos: `free(m)`?



# Desalocando uma Matriz Dinâmica

## Maneira Correta

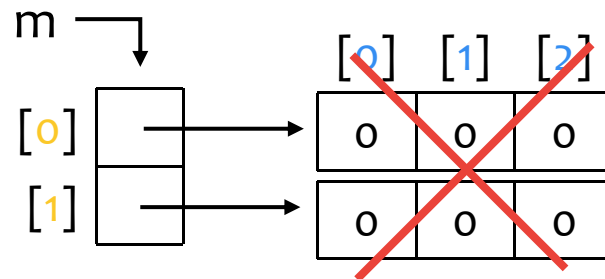
```
for (int i = 0; i < nrows; i++) {  
    free(m[i]);  
}  
free(m);  
m = NULL; // boa prática
```



# Desalocando uma Matriz Dinâmica

Maneira Correta

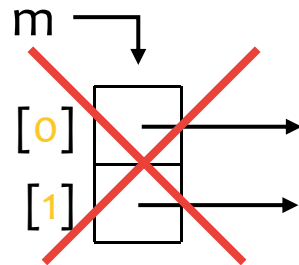
```
→ [ for (int i = 0; i < nrows; i++) {  
    |     free(m[i]);  
    | }  
    | free(m);  
    | m = NULL; // boa prática
```



# Desalocando uma Matriz Dinâmica

## Maneira Correta

```
for (int i = 0; i < nrows; i++) {  
    free(m[i]);  
}  
→ free(m);  
m = NULL; // boa prática
```




# Desalocando uma Matriz Dinâmica

## Maneira Correta

```
for (int i = 0; i < nrows; i++) {  
    free(m[i]);  
}  
free(m);  
→ m = NULL; // boa prática
```

m





Codifique uma função para:

- Alocar uma **matriz dinâmica**
- Imprimir os elementos de uma matriz (*in place*) e seus endereços de memória
- Adicionar um escalar a uma matriz
- Desalocar uma matriz, atribuindo valor NULL ao ponteiro

# Layout de Dados: Row- and Column-Major Order

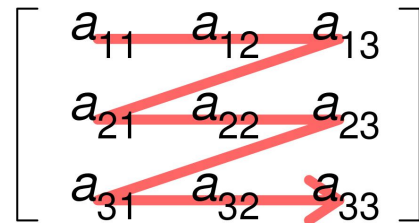
São estratégias para armazenar **arrays multidimensionais** de forma linear na **Memória RAM** —> **layout de dados**.

É importante saber o **layout dos dados** usado para passar corretamente arrays entre programas escritos em diferentes linguagens de programação.

É também importante por questões de desempenho/performance:

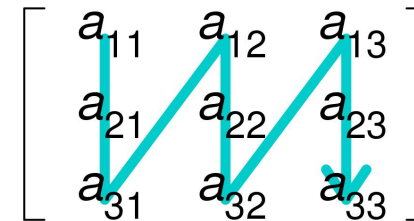
- CPUs processam **dados sequencias mais eficientemente do que dados não sequenciais** --> **CPU Caching**.

Row-major order



C/C++

Column-major order



Fortran, Matlab, R

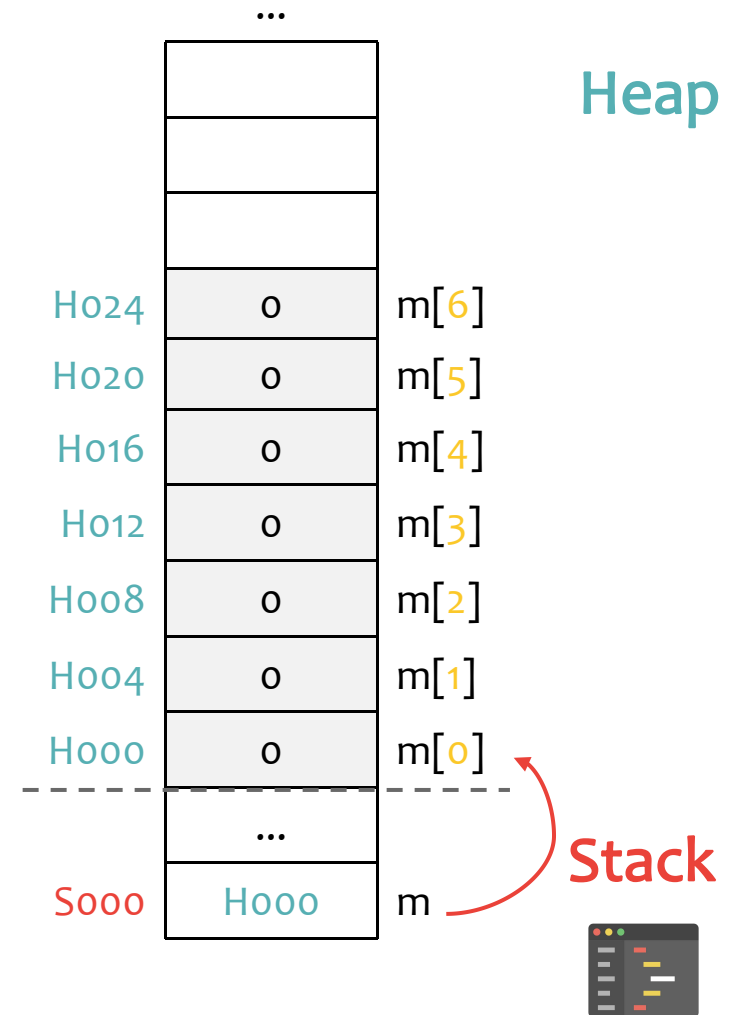


# Alocação Dinâmica: Vetor como Matriz

- Para garantir uma **matriz dinâmica** com elementos **contíguos** na memória RAM, podemos tratar um **vetor como matriz**;

```
int *m = (int*) calloc(nrows*ncols, sizeof(int));
```

- Mas, **não** é mais possível acessar os elementos pelos índices de **linha** e **coluna**
- Então, como acessar o elemento `[i][j]`?



# Dominando Estruturas de Dados 1

## Matrizes Estáticas e Dinâmicas

Prof. Samuel Martins (Samuka)  
@xavecoding @hisamuka

