



## **Aula: Alocação Dinâmica**

### **Introdução a Programação**

---

**Túlio Toffolo & Puca Huachi**  
<http://www.toffolo.com.br>

Departamento de Computação  
Universidade Federal de Ouro Preto

# Aula: Alocação Dinâmica

- 1 Alocação dinâmica
- 2 Alocação dinâmica: erros comuns
- 3 Matrizes dinâmicas
- 4 Exercícios

# Aula: Alocação Dinâmica

- 1 Alocação dinâmica
- 2 Alocação dinâmica: erros comuns
- 3 Matrizes dinâmicas
- 4 Exercícios

# Alocação dinâmica

Comando `malloc`:

- Faz parte da biblioteca `<stdlib.h>`.
- Aloca dinamicamente um bloco consecutivo de *bytes* na memória e retorna o endereço deste bloco.
- Isto permite escrever programas mais flexíveis.
- Exemplo de uso: alocar um vetor de tamanho definido pelo usuário...

# Alocação dinâmica

Uso do método `malloc` para criar um `double`:

```
1 // aloca memória de forma dinâmica
2 double *nro = malloc(sizeof(double));
3
4 // altera o conteúdo da memória apontada por nro para 3.5
5 *nro = 3.5;
6
7 printf("Endereço de memória: %p\n", nro);
8 printf("Valor na memória: %lf\n", *nro);
```

- Este código imprimirá, por exemplo (arquitetura 64 bits):

```
1 Endereço de memória: 0x7feaf4400690
2 Valor na memória: 3.500000
```

# Alocação dinâmica

Uso do método `malloc` para criar um bloco com 3 doubles:

```
1 // aloca memória de forma dinâmica e inicializa com valor 10.5
2 double *nro = malloc(3 * sizeof(double));
3
4 // alterando valores
5 nro[0] = 1.1;
6 nro[1] = 1.5;
7 nro[2] = 2.2;
8
9 for (int i = 0; i < 3; i++)
10     printf("%.11f ", nro[i]);
```

- Este código imprimirá:

```
1 1.1 1.5 2.2
```

## Exemplo de alocação dinâmica

Endereço Conteúdo Nome

0x1000		
0x1004		
0x1008	0x0000	a
0x1012		
0x1016		
0x1020		
0x1024		
0x1028		
0x1032		
0x1036		
0x1040		
0x1044		
0x1048		
0x1052		
0x1056		
0x1060		

```
1  int main() {  
2    → int *a = NULL;  
3    a = malloc(6 * sizeof(int));  
4    for (int i = 0; i < 6; i++)  
5        a[i] = i;  
6    imprimeVetor3(a, 6);  
7    ...  
8 }
```

- Cria o ponteiro **a** com valor inicial **NULL** (0).

## Exemplo de alocação dinâmica

Endereço	Conteúdo	Nome
0x1000		
0x1004		
0x1008	0x1028	a
0x1012		
0x1016		
0x1020		
0x1024		
0x1028		Vetor dinâmico  a
0x1032		
0x1036		
0x1040		
0x1044		
0x1048		
0x1052		
0x1056		
0x1060		

```
1  int main() {  
2      int *a = NULL;  
3      → a = malloc(6 * sizeof(int));  
4      for (int i = 0; i < 6; i++)  
5          a[i] = i;  
6      imprimeVetor3(a, 6);  
7      ...  
8  }
```

- Aloca um bloco de memória com 6 inteiros (usando o comando `malloc`)
- Armazena o endereço de memória no ponteiro `a`.



## Exemplo de alocação dinâmica

Endereço	Conteúdo	Nome
0x1000		
0x1004		
0x1008	0x1028	a
0x1012		
0x1016		
0x1020		
0x1024		
0x1028	0	Vetor dinâmico  a
0x1032	1	
0x1036	2	
0x1040	3	
0x1044	4	
0x1048	5	
0x1052		
0x1056		
0x1060		

```
1  int main() {  
2      int *a = NULL;  
3      a = malloc(6 * sizeof(int));  
4      → for (int i = 0; i < 6; i++)  
5          → a[i] = i;  
6          imprimeVetor3(a, 6);  
7      ...  
8  }
```

- Altera o valor de `a[0]`, `a[1]`, ..., `a[5]`

## Muito importante: liberar a memória

- A memória alocada de forma estática pelo compilador é liberada automaticamente.
- Quando fazemos alocação dinâmica, **liberar a memória se torna nossa responsabilidade**.
- Em C usamos o procedimento `free`
- Exemplo (1):

```
1  int *a = malloc(sizeof(int));  
2  ...  
3  free(a);
```

## Muito importante: liberar a memória

- A memória alocada de forma estática pelo compilador é liberada automaticamente.
- Quando fazemos alocação dinâmica, **liberar a memória se torna nossa responsabilidade**.
- Em C usamos o procedimento `free`
- Exemplo (2):

```
1  int *a = malloc(100 * sizeof(int));  
2  ...  
3  free(a);
```

```

1  #include <stdlib.h>
2  #include <stdio.h>
3
4  void leVetor(int *v, int n) {
5      for (int i = 0; i < n; i++)
6          scanf("%d", &v[i]);
7  }
8
9  int *maior(int *v, int n) {
10     int *maior = v;
11     for (int i = 1; i < n; i++)
12         if (v[i] > *maior)
13             maior = v + i;
14     return maior;
15 }
16
17 int main() {
18     int n, *v;
19     scanf("%d", &n);
20     v = malloc(n * sizeof(int));
21     leVetor(v, n);
22     int *valor = maior(v, n);
23     printf("Maior = %d\n", *valor);
24     free(v);
25     return 0;
26 }

```

# Aula: Alocação Dinâmica

- 1 Alocação dinâmica
- 2 Alocação dinâmica: erros comuns**
- 3 Matrizes dinâmicas
- 4 Exercícios

# Alocação dinâmica: erros comuns

Qual o erro no código abaixo?

```
1  int main()
2  {
3      int n;
4      printf("Qual o tamanho do vetor? ");
5      scanf("%d", &n);
6
7      int *vetor = malloc(n * sizeof(int));
8
9      usaVetor(vetor, n); // função que faz algum uso do vetor...
10
11     printf("Qual o tamanho do segundo vetor? ");
12     scanf("%d", &n);
13     vetor = malloc(n * sizeof(int));
14
15     usaVetor(vetor, n); // função que faz o segundo uso do vetor...
16
17     free(vetor);
18     return 0;
19 }
```

## Qual o erro no código abaixo?

```
1  int *criaPreencheVetor(int tamanho)
2  {
3      int *vetor = malloc(tamanho * sizeof(int));
4      for (int i = 0; i < tamanho; i++) {
5          vetor[i] = i;
6      }
7
8      return vetor;
9  }
10
11 int main()
12 {
13     int n;
14     printf("Qual o tamanho do vetor? ");
15     scanf("%d", &n);
16
17     int *vetor = malloc(n * sizeof(int));
18     vetor = criaPreencheVetor(n);
19
20     free(vetor);
21     return 0;
22 }
```

# Aula: Alocação Dinâmica

- 1 Alocação dinâmica
- 2 Alocação dinâmica: erros comuns
- 3 Matrizes dinâmicas**
- 4 Exercícios



# Matrizes dinâmicas

Matrizes dinâmicas são, na verdade, vetores de vetores!

- Criamos um vetor de ponteiros
- Criamos, para cada ponteiro, um vetor!

# Matrizes dinâmicas

Exemplo:

```
1 // função que cria uma matriz de tamanho n x m
2 int ** criaMatriz(int n, int m)
3 {
4     int **matriz;
5     matriz = malloc(n * sizeof(int*));
6     for (int i = 0; i < n; i++) {
7         matriz[i] = malloc(m * sizeof(int));
8     }
9     return matriz;
10 }
```

Mas... como liberar a memória alocada para a matriz?

# Matrizes dinâmicas

Como liberar a memória alocada por uma matriz?

- Lembre-se: um `free` para cada `malloc`!

Exemplo:

```
1 void liberaMatriz(int **matriz, int n, int m)
2 {
3     for (int i = 0; i < n; i++)
4         free(matriz[i]);
5     free(matriz);
6 }
```

- Obs: note que a variável `m` não é necessária!

## Erros comuns

O que está errado no código abaixo?

```
1 void liberaMatriz(int **matriz, int n, int m)
2 {
3     free(matriz);
4     for (int i = 0; i < n; i++)
5         free(matriz[i]);
6 }
```

Note que neste contexto a **a ordem** faz toda a diferença!

- Após liberar a memória apontada por `matriz`, não podemos mais acessar `matriz[i]`.

# Aula: Alocação Dinâmica

- 1 Alocação dinâmica
- 2 Alocação dinâmica: erros comuns
- 3 Matrizes dinâmicas
- 4 Exercícios**

# Exercícios

## Exercício 1

Crie uma função que:

- 1 recebe um vetor `v` e seu tamanho `n` por parâmetro;
- 2 cria um novo vetor por alocação dinâmica, preenchendo-o com o conteúdo de `v` em ordem inversa;
- 3 retorna este novo vetor.

Dica: utilize o protótipo a seguir:

```
1 int *inverso(int *v, int n);
```

- Crie um exemplo de utilização desta função no método `main()`.
- Não se esqueça de liberar memória!

## Exercícios

### Exercício 2

Defina uma função que retorne a transposta de uma matriz  $m$  com  $\text{lins} \times \text{cols}$  elementos. Utilize o protótipo a seguir:

```
1 int ** transposta(int **m, int lins, int cols);
```

Dica: dada uma matriz  $A$  de tamanho  $n \times m$ , lembre-se que sua transposta  $T$  tem tamanho  $m \times n$  e cada célula  $T_{i,j} = A_{j,i}$ .



Perguntas?