Aula 6: Listas Encadeadas Simples
Linked Lists

DCC302-Estrutura de Dados I

Prof. Me. Acauan C. Ribeiro

1º Mandamento de Estruturas de Dados

"Sempre desenhar a estrutura de dados antes de implementá-la, você deve."

Temos um problema com vetores/arrays

- Suponha que você queira armazenar uma lista de alunos
- Sabemos que, inicalmente, a quantidade de alunos é 5;
- Já sabemos **alocação dinâmica**, podemos utiliza-la:

```c
typedef struct aluno {
  char nome[100];
  int idade;
} Aluno;
```

```c
Aluno **vetAlunos = malloc(5 * sizeof(Aluno *));

for (int i = 0; i < 5; i++) {
  vetAlunos[i] = cria_aluno();
}
```

vetAlunos =

Temos um problema com vetores/arrays

- Precisamos adicionar um **novo aluno**.
- O vetor atual não tem espaço suficiente, Como proceder?

?

Temos um problema com vetores/arrays

- **Opção 1**: Utilizar o ***realloc***, que realoca um espaço de memória
  - Quando mal utilizado acarreta perda de dados.
  - NÃO INDICADO!

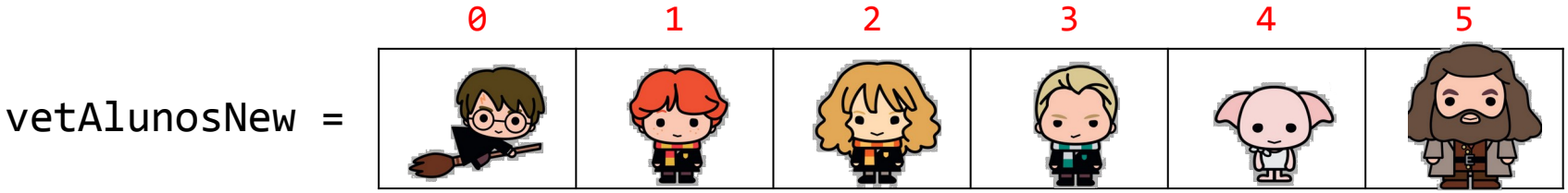
?

Temos um problema com vetores/arrays

- **Opção 2**: Criar um **vetor novo** com **6 posições** e **copiar todos** os valores do antigo vetor e **adicionar** o novo aluno.

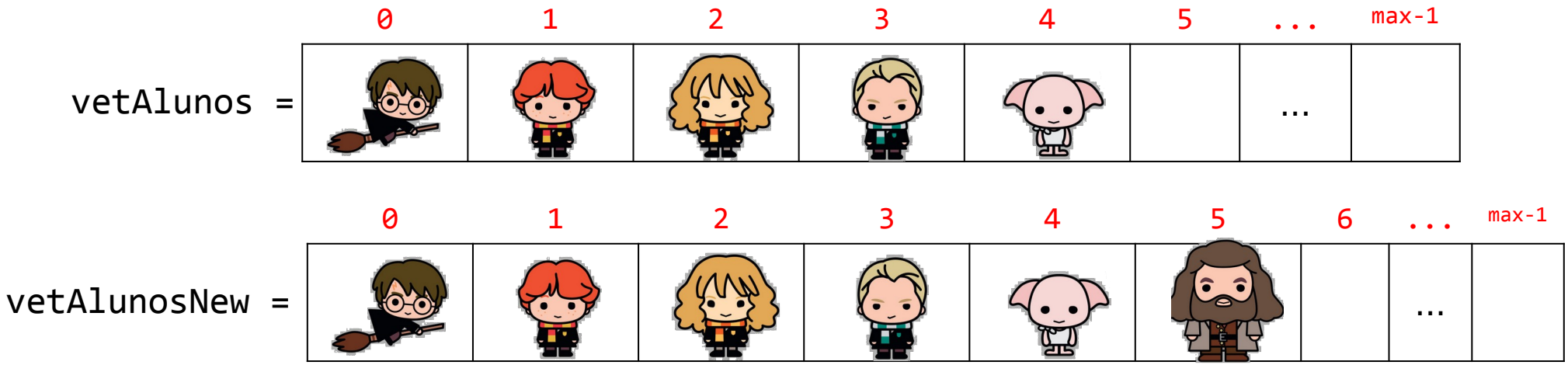
Temos um problema com vetores/arrays

- **Opção 2**: Criar um **vetor novo** com **6 posições** e **copiar todos** os valores do antigo vetor e **adicionar** o novo aluno.
  - Problemas?
    - A cada novo aluno, precisamos alocar um novo vetor;
    - Copiar os dados de um vetor a outro.. custo alto, **ineficiente**.



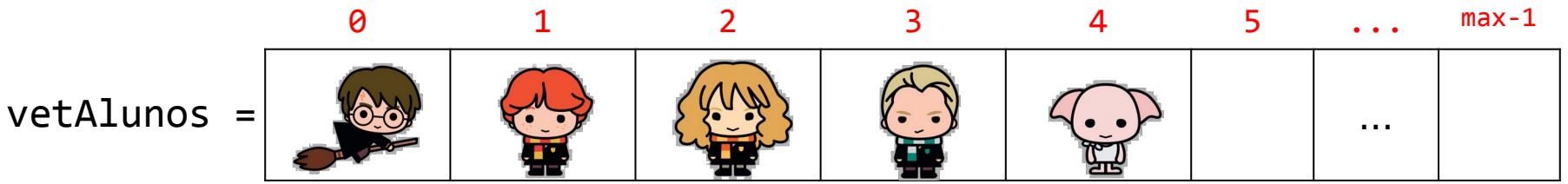
Temos um problema com vetores/arrays

- **Opção 3**: Criar um vetor muito grande, com muitas posições, de modo que "sempre" teremos espaço para inserir um novo aluno;

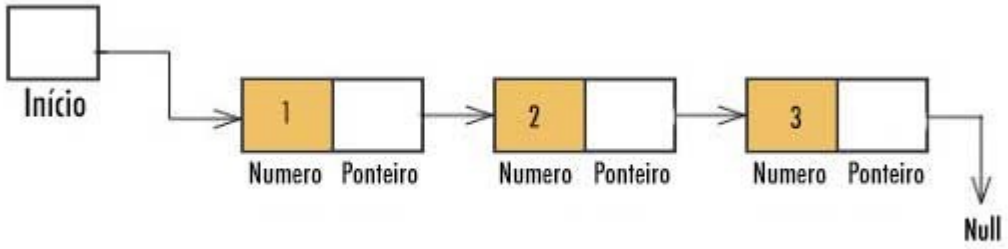


Temos um problema com vetores/arrays

- **Opção 3**: Criar um vetor muito grande, com muitas posições, de modo que "sempre" teremos espaço para inserir um novo aluno;
  - Problemas: Quando chegar ao max; Inserção no início; Remoção (deslocamento)

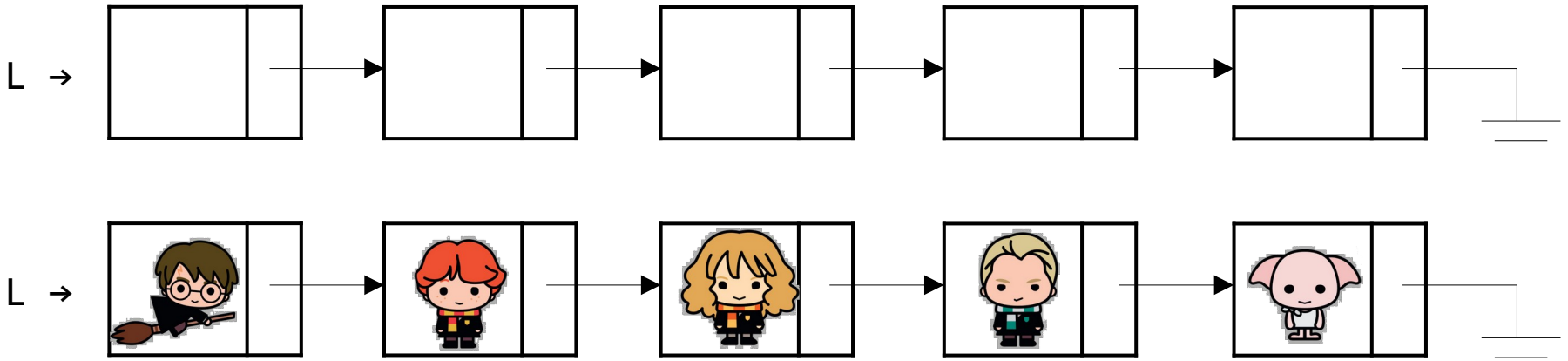


LISTAS ENCADEADAS (LIGADAS)



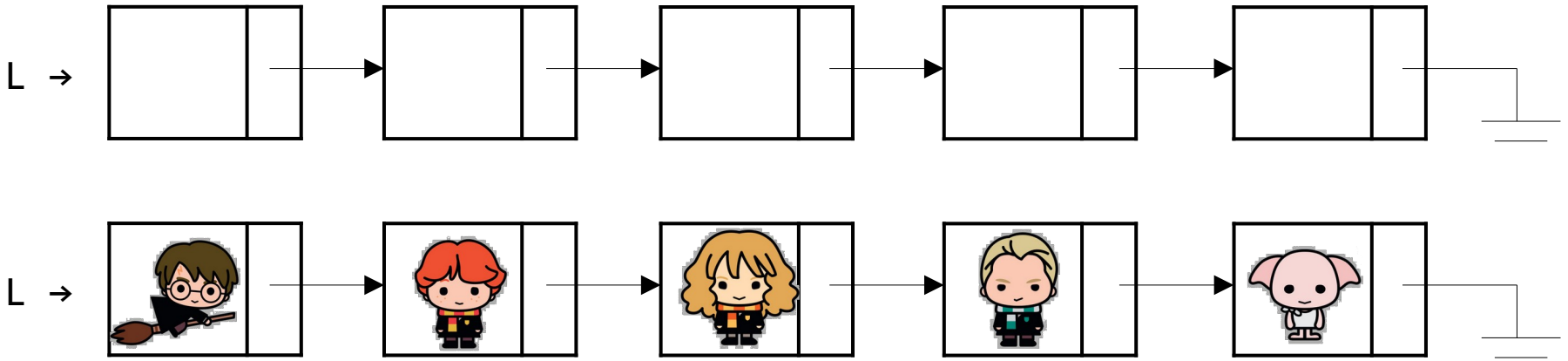
Lista Encadeada

- Uma **lista encadeada (ligada)** é uma representação de uma **sequência de elementos/objetos** na memória do computador;
- Os elementos, **nós da lista**, são armazenados em posições quaisquer da memória e são ligados por **ponteiros**;
  - Logo, **elementos consecutivos** da lista **não** ficam necessariamente em **posições consecutivas na memória**;



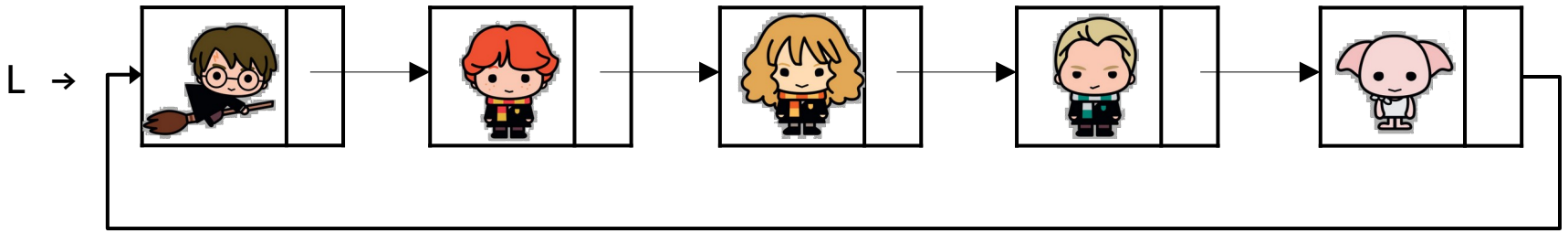
Lista Encadeada

- Cada **nó** contém um **elemento/objeto** de determinado tipo e um **ponteiro para o próximo elemento da lista** --> **Lista Encadeada Simples;**
- No caso do **último nó**, este ponteiro aponta para **NULL**;



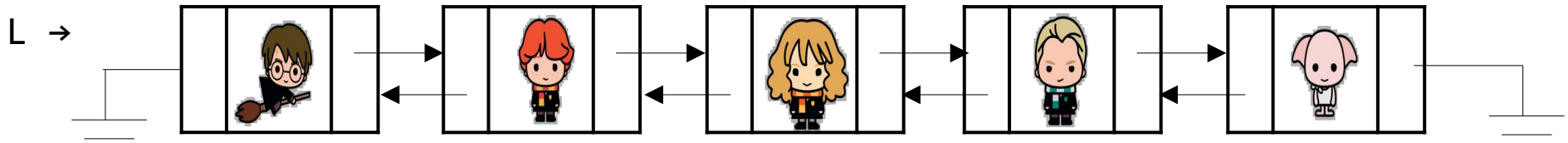
Listas Encadeadas (Variações)

- **Podemos ter ainda outros tipos de listas:**
  - Listas Circulares;



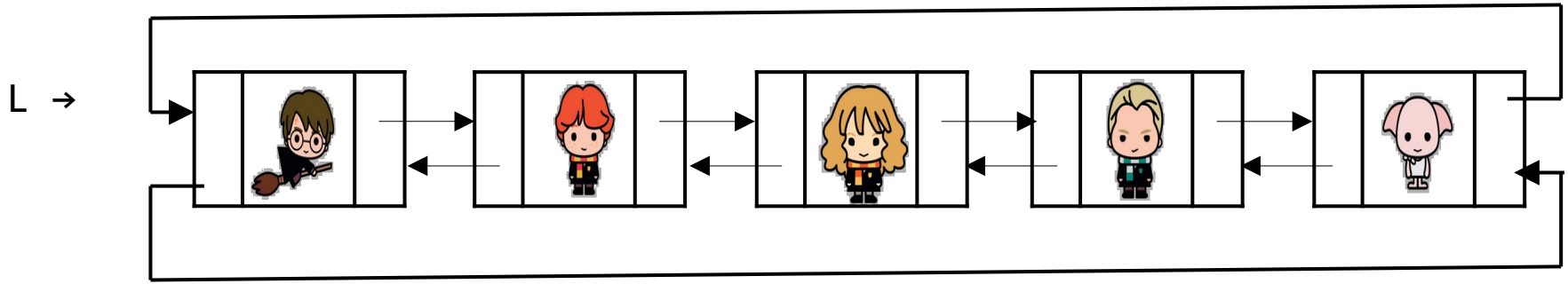
Listas Encadeadas (Variações)

- **Podemos ter ainda outros tipos de listas:**
  - Listas Circulares;
  - Listas Duplamente Encadeadas;

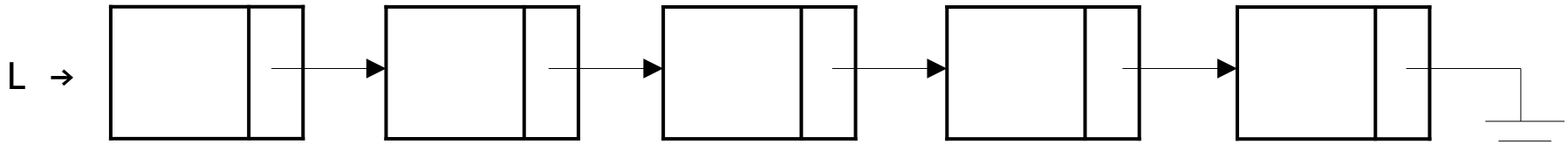


Listas Encadeadas (Variações)

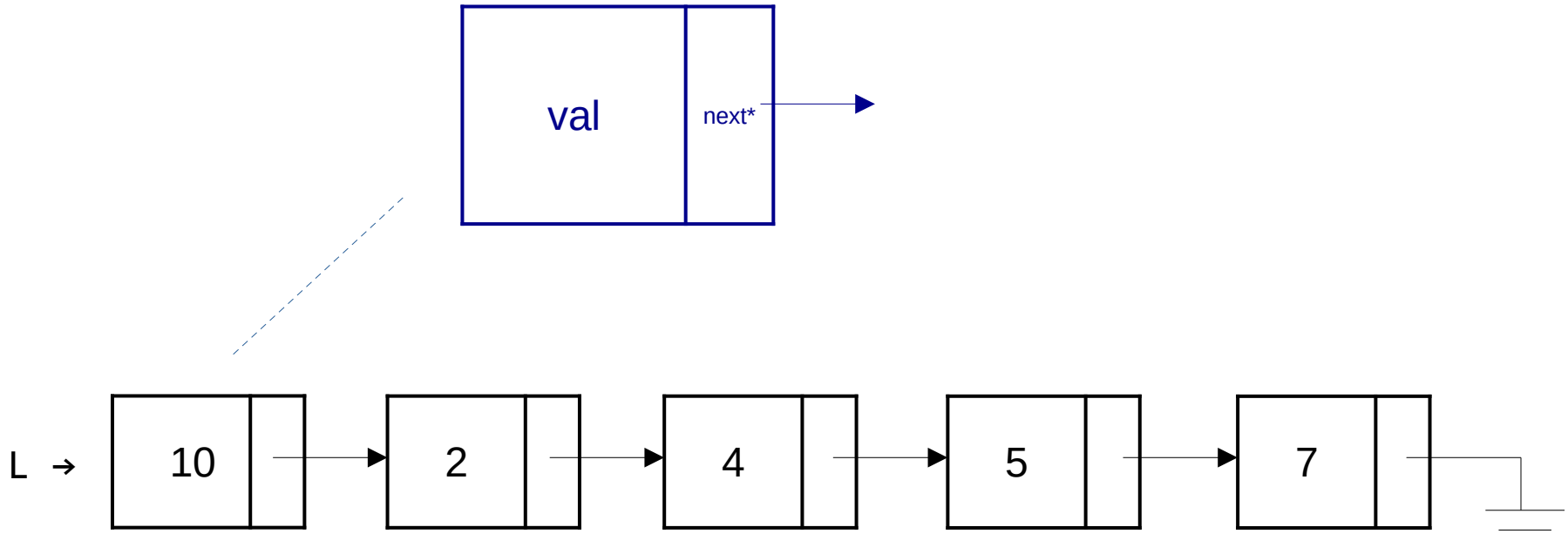
- **Podemos ter ainda outros tipos de listas:**
  - Listas Circulares;
  - Listas Duplamente Encadeadas;
  - Listas Circulares Duplamente Encadeadas;



LISTAS ENCADEADAS SIMPLES



Listas Encadeadas Simples



Lista Encadeada Simples

Ver: https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html (Para se basear na nomenclatura dos métodos)

- Inserção na lista; (add) – inserir elemento na cauda (fim)
- Impressão dos Elementos da Lista
- Inserção na cabeça (início) da lista;
- Remover elementos da lista;
- Contar o número de elementos da Lista;
- Verificar se a lista está vazia e retornar verdadeiro/falso
- Retornar o primeiro elemento;
- Retornar o último elemento;
- Retornar um elemento na posição i