Listas Simplesmente Encadeadas

Prof. Vinicius Ramos

Adaptado de Prof. Cristian Cechinel

Conteúdo

Visão Geral

1. Definição
2. Implementações

01 Definição

O que precisamos saber para poder implementar?

Animação

- https://visualgo.net/pt/list

- Inserção e Remoção
- http://www.algomation.com/algorithm/single-linked-list-insert-delete

Definição

Uma lista encadeada é "*uma estrutura de dados recursiva que é ou vazia (nula) ou uma referência para um nodo possuindo um item genérico e uma referência para uma lista encadeada*" (Sedgewick, 2011)

Definição

O *nodo* é uma entidade abstrata que pode armazenar qualquer tipo de dados além da referência ao próximo nodo que que caracteriza o seu papel na construção da lista encadeada
(Sedgewick, 2011)

Lista Simplesmente Encadeada

Arranjo da memória de uma lista encadeada

- O Nodo possui dois atributos: um item (dado que pode ou não ser parametrizado) e um outro Nodo.

- Define-se o Nodo dentro da classe em que se deseja utilizá-lo e tornamos o mesmo privado, pois ele não é destinado para ser usado pelos clientes *(Sedgewick, 2011)*

Lista Simplesmente Encadeada

Arranjo da memória de uma lista encadeada

- Como com qualquer tipo, criamos um objeto do tipo **Nodo** invocando o **construtor** *new Nodo( )* (com ou sem argumento dependendo das definições do construtor) *(Sedgewick, 2011)*

Lista Simplesmente Encadeada

Arranjo da memória de uma lista encadeada

- Numa lista encadeada, para cada novo elemento inserido na estrutura, alocamos um espaço de memória para armazená-lo. Desta forma, o espaço total de memória gasto pela estrutura é proporcional ao número de elementos nela armazenado.

Lista Simplesmente Encadeada

Arranjo da memória de uma lista encadeada

- Como não podemos garantir que os elementos armazenados na lista ocuparão um espaço de memória contíguo, não possuímos um acesso direto aos elemntos da lista *(Celes, Serqueira e Rangel, 2004)*

Lista Simplesmente Encadeada

Arranjo da memória de uma lista encadeada

- Em outras linguagens, além de alocar dinamicamente os dados na memória, é necessário, também, liberar os espaços que não estão mais sendo utilizados
- Como já sabemos, programas em Java não liberam explicitamente memória alocada dinamicamente.
- O Java utiliza uma coleta de lixo automática sobre objetos aos quais não se faz mais referência em um programa

Lista Simplesmente Encadeada

Arranjo da memória de uma lista encadeada

- Para percorrer os elementos da lista é necessário guardar o encadeamento dos mesmos.
- Isso é realizado por meio da referência (ponteiro) ao próximo elemento da lista (do mesmo tipo da própria classe que o está referenciando) (Celes, Serqueira e Rangel, 2004)

Lista Simplesmente Encadeada

Arranjo da memória de uma lista encadeada

- Para percorrer os elementos da lista é necessário guardar o encadeamento dos mesmos.
- Isso é realizado por meio da referência (ponteiro) ao próximo elemento da lista (do mesmo tipo da própria classe que o está referenciando) (Celes, Serqueira e Rangel, 2004)

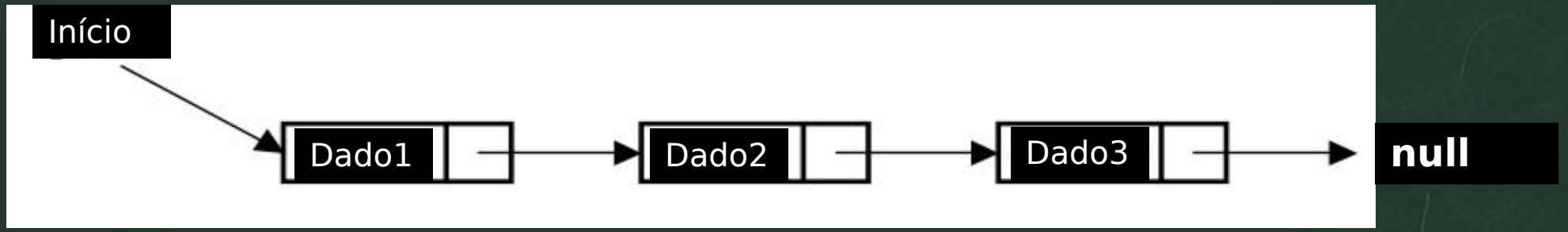
Lista Simplesmente Encadeada

Arranjo da memória de uma lista encadeada

- Para percorrer os elementos da lista é necessário guardar o encadeamento dos mesmos.
- Isso é realizado por meio da referência (ponteiro) ao próximo elemento da lista (do mesmo tipo da própria classe que o está referenciando) (Celes, Serqueira e Rangel, 2004)

Lista Simplesmente Encadeada

Arranjo da memória de uma lista encadeada



Adaptado de (CELES, CERQUEIRA, e RANGEL, 2004)

Referências

- CELES, W., R. CERQUEIRA, and JL RANGEL. Introdução a Estruturas de Dados. Rio de Janeiro, RJ, Brasil: Campus, 2004. 294 p. ISBN 85-352-1228-0.

- Sedgewick, Robert, and Kevin Wayne. Algorithms. Addison-Wesley Professional, 2011.

OBRIGADO!

Listas Simplesmente Encadeadas na Prática

Prof. Vinicius Ramos

Adaptado de Prof. Cristian Cechinel

Visão Geral

Conteúdo

02 Implementações

Vamos colocar a mão na massa!!!

Lista Simplesmente Encadeada

- Iniciamos com uma classe aninhada que define a abstração do Nodo

```
16      private class Nodo {
17          int dado;
18          Nodo proximo;
19      }
```

Nodo de Inteiros

```
16      private class Nodo {
17          Item dado;
18          Nodo proximo;
19      }
```

Nodo Genérico

Lista Simplesmente Encadeada

- A lista encadeada terá como atributo um ponteiro para o primeiro Nodo da lista (inicio) que por sua vez é inicialmente nulo (null)

```
13    public class ListaEncadeada {
14        private Nodo inicio;
15
16        private class Nodo {
17            int dado;
18            Nodo proximo;
19        }
20
21        public ListaEncadeada(){
22            inicio = null;
23        }
```

Inserção no início da lista

Listas Simplesmente Encadeadas na Prática

Prof. Vinicius Ramos

Adaptado de Prof. Cristian Cechinel

Lista Simplesmente Encadeada

Inserção de um novo Nodo no início da lista encadeada

1. A inserção de um novo Nodo na lista deve alocar dinamicamente o espaço para armazenar esse novo nodo, guardar a informação no novo nodo e fazer este nodo apontar para o próximo elemento.

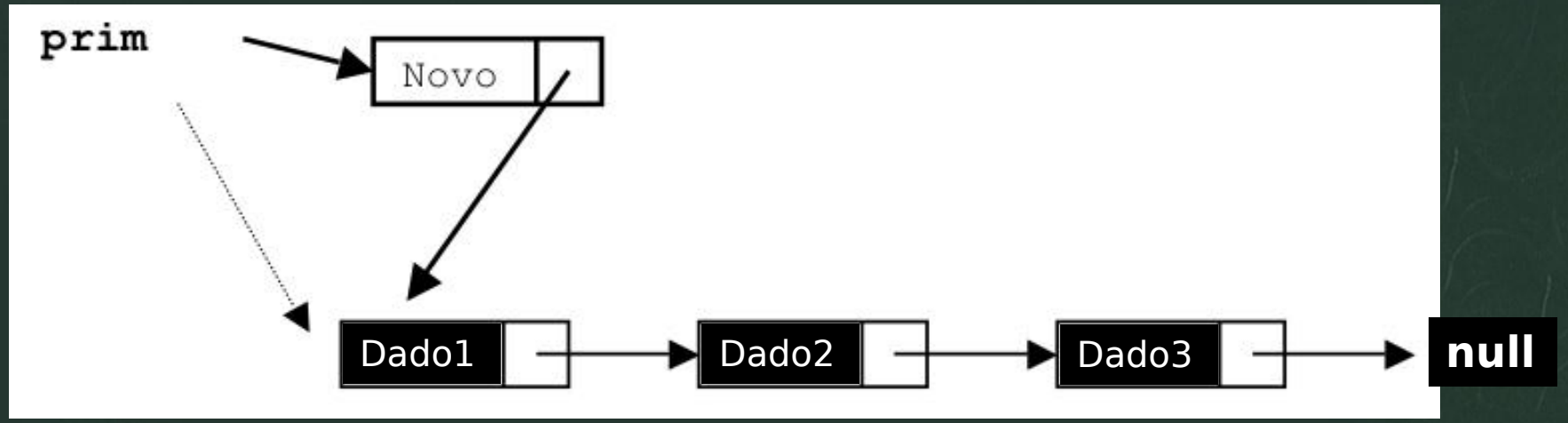
2. Quando estamos inserindo no início da lista, o próximo elemento é o primeiro elemento da lista.

3. E o início da lista passa a ser o novo elemento

https://visualgo.net/pt/list

Lista Simplesmente Encadeada

Inserção de um novo Nodo no início da lista encadeada



Adaptado de (CELES, CERQUEIRA, e RANGEL, 2004)

Lista Simplesmente Encadeada

Inserção de um novo Nodo no início da lista encadeada

```
25      public void insereInicio(int n){
26          Nodo novo = new Nodo();
27          novo.dado = n;
28          novo.proximo = inicio;
29          inicio = novo;
30      }
```

Impressão dos elementos da lista

Listas Simplesmente Encadeadas na Prática

Prof. Vinicius Ramos

Adaptado de Prof. Cristian Cechinel

Lista Simplesmente Encadeada

Percorrimento da lista para impressão

O percorrimento de uma lista com vetores é realizado da seguinte maneira

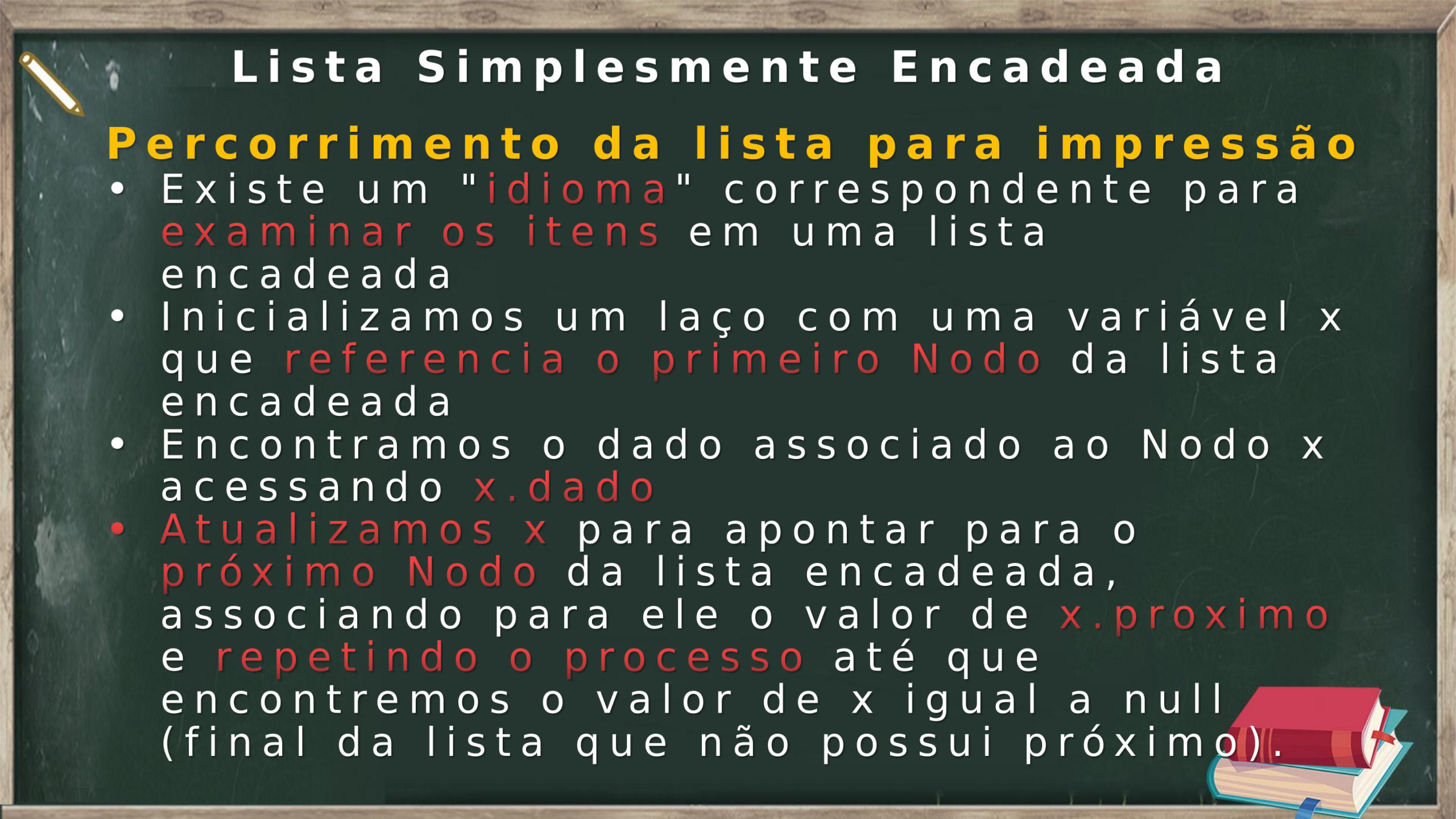
Laço para processar os itens de um vetor a[ ]:

```
for (int i = 0; i < N; i++) {
        // Processa a[i]
}
```

Lista Simplesmente Encadeada

Percorrimento da lista para impressão

- Existe um "idioma" correspondente para examinar os itens em uma lista encadeada
- Inicializamos um laço com uma variável x que referencia o primeiro Nodo da lista encadeada
- Encontramos o dado associado ao Nodo x acessando x.dado
- Atualizamos x para apontar para o próximo Nodo da lista encadeada, associando para ele o valor de x.proximo e repetindo o processo até que encontremos o valor de x igual a null (final da lista que não possui próximo).



Lista Simplesmente Encadeada

Percorrimento da lista para impressão

```
for (Nodo x = inicio; x != null; x = x.proximo) {
        // Processa x.dado
}
```

Lista Simplesmente Encadeada

Percorrimento da lista para impressão

```
32      public void imprimeLista(){
33          for (Nodo x = inicio; x!= null; x=x.proximo){
34              System.out.print(x.dado+"->");
35          }
36
37          System.out.println();
38      }
```

Lista Simplesmente Encadeada

Percorrimento da lista para impressão

- Código usando um *WHILE*

```
32      public void imprimeLista(){
33          Nodo temp = inicio;
34          while (temp != null){
35              System.out.print(temp.dado+"->");
36              temp = temp.proximo;
37          }
38          System.out.println();
39      }
```

Referências

- CELES, W., R. CERQUEIRA, and JL RANGEL. Introdução a Estruturas de Dados. Rio de Janeiro, RJ, Brasil: Campus, 2004. 294 p. ISBN 85-352-1228-0.

- Sedgewick, Robert, and Kevin Wayne. Algorithms. Addison-Wesley Professional, 2011.

OBRIGADO!

Remoção no início da lista

Listas Simplesmente Encadeadas na Prática

Prof. Vinicius Ramos

Adaptado de Prof. Cristian Cechinel

Lista Simplesmente Encadeada

Remoção do primeiro Nodo da lista

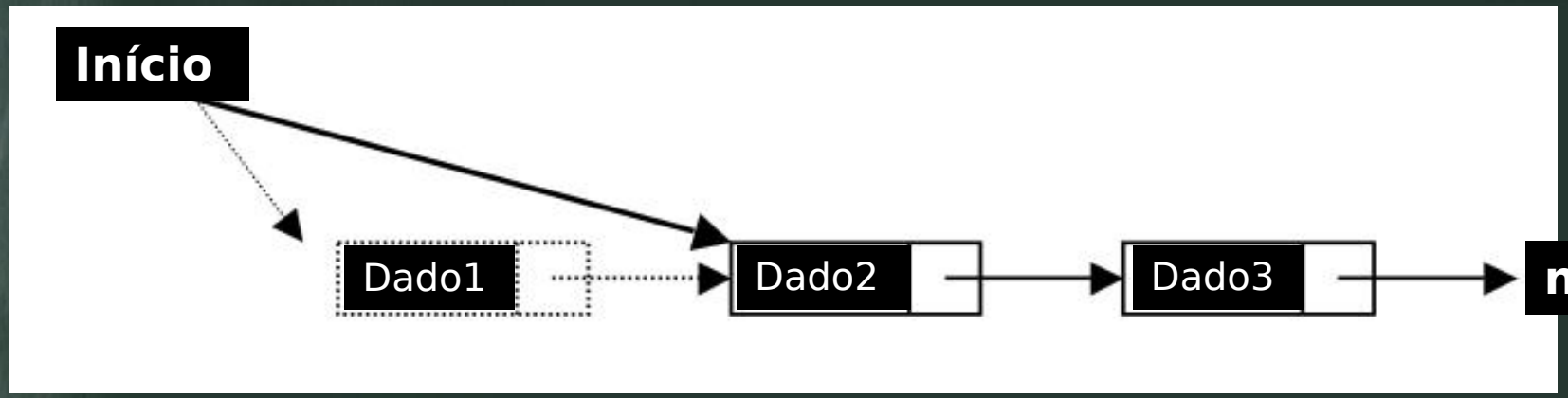
1. A remoção do primeiro Nodo da lista implica em guardar a informação do primeiro Nodo da lista (caso a mesma não seja vazia)

2. E atualizar o início da lista, que passa a ser o próximo elemento

https://visualgo.net/pt/list

Lista Simplesmente Encadeada

Remoção do primeiro Nodo da lista



Adaptada de (Celes, Serqueira e Rangel, 2004)

Lista Simplesmente Encadeada

Remoção do primeiro Nodo da lista

```
70      public Integer retiraInicio(){
71          if (inicio != null){
72              Nodo retirado = inicio;
73              inicio = inicio.proximo;
74              return retirado.dado;
75          }
76          else {
77              return null;
78          }
79      }
```

Lista Simplesmente Encadeada

Remoção do Nodo no meio da lista

1. A remoção do um nodo no meio da lista implica em encontrar o elemento que se deseja remover, sempre garantindo que a lista não esteja inicialmente vazia.

2. Ao encontrar o elemento a ser removido, deve-se guardar a informação para retorno e atualizar os ponteiros da lista.

3. O Nodo anterior ao Nodo removido deve agora apontar para o próximo Nodo daquele que foi removido

https://visualgo.net/pt/list

Remoção no meio da lista

Listas Simplesmente Encadeadas na Prática

Prof. Vinicius Ramos

Adaptado de Prof. Cristian Cechinel

Lista Simplesmente Encadeada

Remoção do Nodo no meio da lista

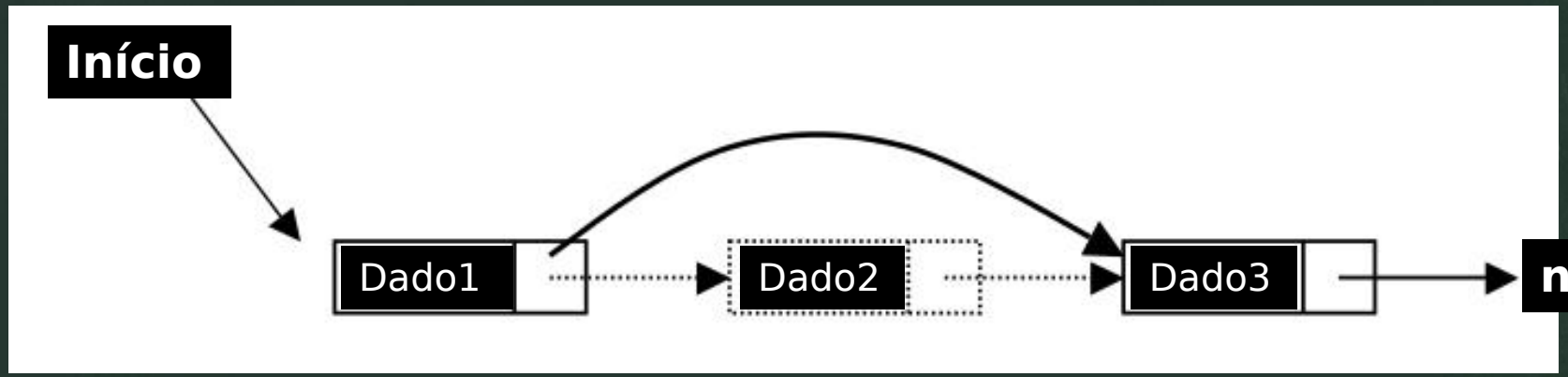
4. Para isso, ao longo da busca do elemento, é necessário sempre guardar uma cópia do Nodo anterior, de maneira que ao final tenhamos acesso ao ponteiro que deverá ser atualizado.

5. Em uma lista encadeada, a remoção (ou a inserção no meio da lista) não implica em deslocamento dos elementos como acontecia na lista representada por vetores.

https://visualgo.net/pt/list

Lista Simplesmente Encadeada

Remoção do Nodo no meio da lista



Adaptada de (Celes, Serqueira e Rangel, 2004)

Lista Simplesmente Encadeada

Remoção do Nodo no meio da lista

```
49      public Integer retiraNodo(Integer n){
50          Nodo temp = this.inicio;
51          Nodo anterior = null;
52          while (temp!= null && temp.dado != n){
53              anterior = temp;
54              temp = temp.proximo;
55          }
56          if (temp == null){//significa que não achou
57              return null;
58          }
59          if (anterior == null) {//encontrou no inicio
60              int retirado = inicio.dado;
61              inicio = inicio.proximo;
62              return retirado;
63          }
64          //encontrou no meio
65          int retirado = temp.dado;
66          anterior.proximo = temp.proximo;
67          return retirado;
68      }
```

Lista Simplesmente Encadeada

Remoção do Nodo no meio da lista

```
49      public Integer retiraNodo(Integer n){
50          Nodo temp = this.inicio;
51          Nodo anterior = null;
52          while (temp!= null && temp.dado != n){
53              anterior = temp;
54              temp = temp.proximo;
55          }
56          if (temp == null){//significa que não achou
57              return null;
58          }
59          if (anterior == null) {//encontrou no inicio
60              int retirado = inicio.dado;
61              inicio = inicio.proximo;
62              return retirado;
63          }
64          //encontrou no meio
65          int retirado = temp.dado;
66          anterior.proximo = temp.proximo;
67          return retirado;
68      }
```

- Guarda ponteiro que irá percorrer a lista
- Guarda ponteiro para o Nodo anterior

Lista Simplesmente Encadeada

Remoção do Nodo no meio da lista

```
49      public Integer retiraNodo(Integer n){
50          Nodo temp = this.inicio;
51          Nodo anterior = null;
52          while (temp!= null && temp.dado != n){
53              anterior = temp;
54              temp = temp.proximo;
55          }
56          if (temp == null){//significa que não achou
57              return null;
58          }
59          if (anterior == null) {//encontrou no inicio
60              int retirado = inicio.dado;
61              inicio = inicio.proximo;
62              return retirado;
63          }
64          //encontrou no meio
65          int retirado = temp.dado;
66          anterior.proximo = temp.proximo;
67          return retirado;
68      }
```

- Percorre a lista até o final (null) ou até o momento em que o conteúdo do nodo for igual ao elemento procurado
- Guarda sempre o ponteiro para o Nodo atual como anterior
- Atualiza o Nodo atual para ser o próximo e segue a busca

Lista Simplesmente Encadeada

Remoção do Nodo no meio da lista

```
49      public Integer retiraNodo(Integer n){
50          Nodo temp = this.inicio;
51          Nodo anterior = null;
52          while (temp!= null && temp.dado != n){
53              anterior = temp;
54              temp = temp.proximo;
55          }
56          if (temp == null){//significa que não achou
57              return null;
58          }
59          if (anterior == null) {//encontrou no inicio
60              int retirado = inicio.dado;
61              inicio = inicio.proximo;
62              return retirado;
63          }
64          //encontrou no meio
65          int retirado = temp.dado;
66          anterior.proximo = temp.proximo;
67          return retirado;
68      }
```

- Se a busca chegou ao final da lista sem encontrar o elemento, o valor da variável temp será nulo

Lista Simplesmente Encadeada

Remoção do Nodo no meio da lista

```
49      public Integer retiraNodo(Integer n){
50          Nodo temp = this.inicio;
51          Nodo anterior = null;
52          while (temp!= null && temp.dado != n){
53              anterior = temp;
54              temp = temp.proximo;
55          }
56          if (temp == null){//significa que não achou
57              return null;
58          }
59          if (anterior == null) {//encontrou no inicio
60              int retirado = inicio.dado;
61              inicio = inicio.proximo;
62              return retirado;
63          }
64          //encontrou no meio
65          int retirado = temp.dado;
66          anterior.proximo = temp.proximo;
67          return retirado;
68      }
```

- Se o anterior é igual a null significa que o código não entrou no laço de repetição, ou seja, que o primeiro Nodo contém o elemento que deve ser retirado (temp.dado == n)
- Deve-se então retirar o Nodo do início

Lista Simplesmente Encadeada

Remoção do Nodo no meio da lista

```
49      public Integer retiraNodo(Integer n){
50          Nodo temp = this.inicio;
51          Nodo anterior = null;
52          while (temp!= null && temp.dado != n){
53              anterior = temp;
54              temp = temp.proximo;
55          }
56          if (temp == null){//significa que não achou
57              return null;
58          }
59          if (anterior == null) {//encontrou no inicio
60              int retirado = inicio.dado;
61              inicio = inicio.proximo;
62              return retirado;
63          }
64          //encontrou no meio
65          int retirado = temp.dado;
66          anterior.proximo = temp.proximo;
67          return retirado;
68      }
```

- Caso não tenha entrado em nenhuma das condições anteriores, o elemento foi encontrado em algum Nodo do meio da lista.
- O nodo temp deve ser retirado
- O anterior deve apontar para o próximo Nodo após o temp.

Referências

- CELES, W., R. CERQUEIRA, and JL RANGEL. Introdução a Estruturas de Dados. Rio de Janeiro, RJ, Brasil: Campus, 2004. 294 p. ISBN 85-352-1228-0.

- https://inf.ufes.br/~pdcosta/ensino/2012-2-estruturas-de-dados/slides/Aula9(listas).pdf

- Sedgewick, Robert, and Kevin Wayne. Algorithms. Addison-Wesley Professional, 2011.

OBRIGADO!