Listas Simplesmente Encadeadas

Estruturas de Dados e Algoritmos

Listas Simplesmente Encadeadas

- A lista simplesmente encadeada implementada anteriormente possuia apenas uma referência (ponteiro) para o início da lista

Listas Simplesmente Encadeadas

Arranjo da memória de uma lista encadeada

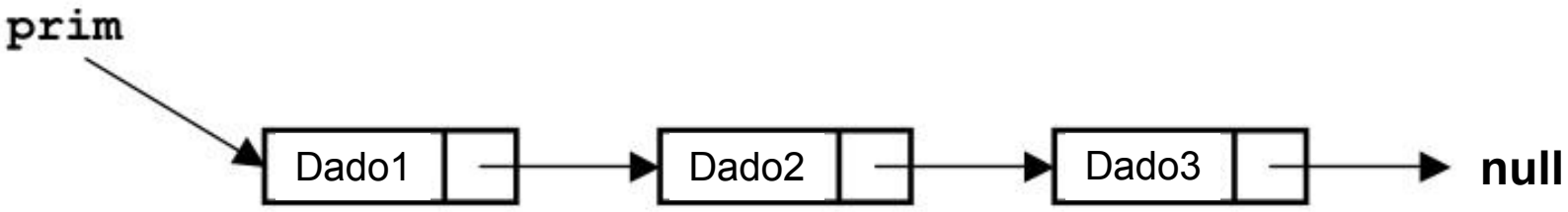

Figura adaptada de (Celes, Serqueira e Rangel, 2004)

Listas Simplesmente Encadeadas

- Dessa maneira, caso desejemos acessar o final da lista, é necessário percorrer toda a lista para conhecer o último elemento

- Uma alternativa para conhecermos diretamente o último elemento da lista é trabalharmos com um ponteiro adicional que aponte para esse elemento

Listas Simplesmente Encadeadas

Arranjo da memória de uma lista encadeada

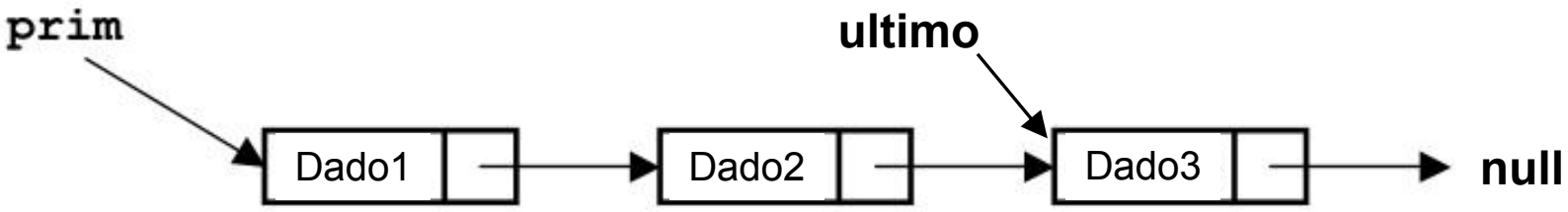

Figura adaptada de (Celes, Serqueira e Rangel, 2004)

Listas Simplesmente Encadeadas

- Nessa nova implementação, a lista encadeada terá como atributo um ponteiro para o primeiro Nodo da lista (inicio) e outro para o final da lista (ultimo) que são inicialmente nulos (null)

```
13  public class ListaEncadeada {
14      private Nodo inicio;
15      private Nodo ultimo;
16
17      private class Nodo {
18          int dado;
19          Nodo proximo;
20      }
21
22      public ListaEncadeada(){
23          inicio = null;
24          ultimo = null;
25      }
```

Listas Simplesmente Encadeadas

- A inserção desse novo ponteiro implica em uma atualização do código, de maneira a manter esse apontamento correto após cada operação realizada na lista

- Ainda, a existência desse ponteiro facilitará, por exemplo, o acesso direto ao último elemento permitindo inclusões no final da lista sem a necessidade de percorrer a lista inteira

Listas Simplesmente Encadeadas

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
4. O ponteiro para o final da lista (último) não é alterado em uma inserção no início, **salvo nos casos em que a lista estava vazia.**

Listas Simplesmente Encadeadas

Inserção de um novo Nodo no início da lista encadeada

ANTES DE INSERIR O PONTEIRO PARA O ÚLTIMO

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

Listas Simplesmente Encadeadas

Inserção de um novo Nodo no início da lista encadeada

APÓS INSERIR O PONTEIRO PARA O ÚLTIMO

```
public void insereInicio(int n){
    Nodo novo = new Nodo();
    novo.dado = n;
    if (this.vazia()){
        novo.proximo = inicio;
        inicio = novo;
        ultimo = inicio;
    }
    else {
        novo.proximo = inicio;
        inicio = novo;
    }
}
```

Listas Simplesmente Encadeadas

Inserção de um novo Nodo no início da lista encadeada

APÓS INSERIR O PONTEIRO PARA O ÚLTIMO

```
public void insereInicio(int n){
    Nodo novo = new Nodo();
    novo.dado = n;
    if (this.vazia()){
        novo.proximo = inicio;
        inicio = novo;
        ultimo = inicio;
    }
    else {
        novo.proximo = inicio;
        inicio = novo;
    }
}
```

Listas Simplesmente Encadeadas

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
3. Como a remoção está sendo realizada no início, não há alteração no ponteiro para o final da lista (último), **salvo nos casos em que a lista ficar vazia após a remoção, ou seja, quando o inicio e o ultimo estiverem apontando para um mesmo elemento (único elemento da lista)**

Listas Simplesmente Encadeadas

Remoção do primeiro Nodo da lista

ANTES DE INSERIR O PONTEIRO PARA O ÚLTIMO

```
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      }
```

```
75      public Integer retiraInicio(){
76          if (vazia()) return null;
77          else {
78              Nodo retirado = inicio;
79              inicio = inicio.proximo;
80              return retirado.dado;
81          }
82      }
```

Remoção do primeiro Nodo da lista

APÓS INSERIR O PONTEIRO PARA O ÚLTIMO

```
107     public Integer retiraInicio(){
108         if (vazia()) return null;
109         Nodo retirado = inicio;
110         if (inicio == ultimo)
111             ultimo = inicio = null;
112         else
113             inicio = inicio.proximo;
114         return retirado.dado;
115     }
```

Novos métodos

Listas Simplesmente Encadeadas

Inserção no final da lista

1. A inserção de um novo Nodo no final da lista deve alocar dinamicamente o espaço para armazenar esse novo nodo, guardar a informação no novo nodo e fazer este nodo apontar para null (pois o mesmo será o último)
2. O último nodo da lista deve agora apontar para o novo nodo
3. E o ponteiro para o final da lista (último) deve também apontar para o novo nodo

Listas Simplesmente Encadeadas

Inserção no final da lista

4. Com a inserção é feita no final, não é alterado o ponteiro para o início, **salvo nos casos em que a lista estiver vazia**

Listas Simplesmente Encadeadas

Inserção no final da lista

```
35  public void insereFinal(int n){
36      Nodo novo = new Nodo();
37      novo.dado = n;
38      novo.proximo = null;
39      if (vazia()){
40          inicio = novo;
41          ultimo = novo;
42      }
43      else {
44          ultimo.proximo = novo;
45          ultimo = novo;
46      }
```

O novo nodo será o último e o seu próximo é null

Listas Simplesmente Encadeadas

Inserção no final da lista

```
35      public void insereFinal(int n){
36          Nodo novo = new Nodo();
37          novo.dado = n;
38          novo.proximo = null;
39          if (vazia()){
40              inicio = novo;
41              ultimo = novo;
42          }
43          else {
44              ultimo.proximo = novo;
45              ultimo = novo;
46          }
```

Caso a lista esteja inicialmente vazia, ambos inicio e último apontarão para o mesmo elemento

Listas Simplesmente Encadeadas

Inserção no final da lista

```
35      public void insereFinal(int n){
36          Nodo novo = new Nodo();
37          novo.dado = n;
38          novo.proximo = null;
39          if (vazia()){
40              inicio = novo;
41              ultimo = novo;
42          }
43          else {
44              ultimo.proximo = novo;
45              ultimo = novo;
46          }
```

Caso já exista algum elemento na lista é preciso atualizar o ponteiro do último elemento.

O próximo do último para a ser o novo nodo

O último passa a ser o novo nodo.

Listas Simplesmente Encadeadas

Remoção no final da lista encadeada

1. A remoção de um Nodo do final da lista implica em localizar o penúltimo elemento para que o mesmo possa ser apontado como o novo final da lista (último)
2. Após localizar o penúltimo elemento, ele passa a ser o último e apontar para null
3. Como a remoção é no final da lista, não é alterado o ponteiro para o início, **salvo nos casos em que a lista estiver com apenas um único elemento e se tornar vazia.**

Listas Simplesmente Encadeadas

Remoção de um novo Nodo no final da lista encadeada

```
public Integer retiraUltimo(){
    if (vazia()) return null;
    Nodo retirado = ultimo;
    if (inicio == ultimo)
        ultimo = inicio = null;
    else { //localiza o novo último (penultimo)
        Nodo temp = inicio;
        while (temp.proximo != ultimo)
                temp = temp.proximo;
        ultimo = temp;
        temp.proximo = null;
    }

    return retirado.dado;
}
```

Listas Simplesmente Encadeadas

Remoção de um novo Nodo no final da lista encadeada

```
public Integer retiraUltimo(){
    if (vazia()) return null;
    Nodo retirado = ultimo;
    if (inicio == ultimo)
        ultimo = inicio = null;
    else { //localiza o novo último (penultimo)
        Nodo temp = inicio;
        while (temp.proximo != ultimo)
                temp = temp.proximo;
        ultimo = temp;
        temp.proximo = null;
    }

    return retirado.dado;
}
```

A lista possui apenas 1 elemento e ficará vazia

Listas Simplesmente Encadeadas

Remoção de um novo Nodo no final da lista encadeada

```
public Integer retiraUltimo(){
    if (vazia()) return null;
    Nodo retirado = ultimo;
    if (inicio == ultimo)
        ultimo = inicio = null;
    else { //localiza o novo último (penultimo)
        Nodo temp = inicio;
        while (temp.proximo != ultimo)
                temp = temp.proximo;
        ultimo = temp;
        temp.proximo = null;
    }

    return retirado.dado;
}
```

Localiza o penúltimo elemento

Listas Simplesmente Encadeadas

Remoção de um novo Nodo no final da lista encadeada

```
public Integer retiraUltimo(){
    if (vazia()) return null;
    Nodo retirado = ultimo;
    if (inicio == ultimo)
        ultimo = inicio = null;
    else { //localiza o novo último (penultimo)
        Nodo temp = inicio;
        while (temp.proximo != ultimo)
            temp = temp.proximo;
        ultimo = temp;
        temp.proximo = null;
    }

    return retirado.dado;
}
```

O penúltimo passa a ser o último e aponta para null

Listas Simplesmente Encadeadas

Remoção de um novo Nodo no final da lista encadeada

```
public Integer retiraUltimo(){
    if (vazia()) return null;
    Nodo retirado = ultimo;
    if (inicio == ultimo)
        ultimo = inicio = null;
    else { //localiza o novo último (penultimo)
        Nodo temp = inicio;
        while (temp.proximo != ultimo)
                temp = temp.proximo;
        ultimo = temp;
        temp.proximo = null;
    }

    return retirado.dado;
}
```

Retorna o valor armazenado no nodo removido

Exercícios

Exercícios

Utilize a nova estrutura lista para criar as estruturas de dados de uma Fila e uma Pilha

1. A Pilha deve possuir 2 métodos apenas (empilha e desempilha). É possível também inserir os métodos pilhaVazia() e imprimePilha()

2. A Fila deve possuir 2 métodos apenas (insereFila e retiraFila). Lembrando que a inserção acontece no final da fila e a retirada acontece no começo. **É possível também inserir os métodos filaVazia() e imprimeFila()**

Pilha com uso de lista encadeada

Classe Pilha com uso de uma lista encadeada

```
public class PilhaEncadeada {
    private ListaEncadeada lista;

    PilhaEncadeada (){
        this.lista = new ListaEncadeada();
    }

    void adicionaTopo(int item){
        lista.insereFinal(item);
    }

    Integer retiraTopo(){
        return lista.retiraUltimo();
    }
}
```

Referências utilizadas

- 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.
- Deitel, H. M., & Deitel, P. J. (2003). Java, como programar. 4ª Edição. São Paulo.