
Uma empresa deseja recrutar vários funcionários nacionalmente. Ela dispõe de um conjunto de filiais em diferentes regiões metropolitanas que se ocupam de realizar testes com os funcionários locais e de avaliá-los. Cada filial regional leva então os resultados de seus candidatos locais ao escritório nacional que deve finalizar o recrutamento.

Você vai implementar nesta prática um conjunto de programas para permitir à empresa tratar os dados transmitidos por cada uma das filiais regionais e de selecionar os melhores candidatos.

Como o número de candidatos é variável, em função da região e do desenrolar do processo de recrutamento (seleções, desistências, etc.), é preferível utilizar listas encadeadas de candidatos ao invés de arrays. Logo, vamos implementar uma classe `Candidato` para representar os candidatos, depois uma classe `ListaCandidatos` que terá os métodos para testar se uma lista é vazia, calcular seu tamanho, e adicionar ou retirar um elemento.

1. Os candidatos por região e suas notas

Cada região possui um arquivo de seus candidatos locais com suas avaliações, ordenados por ordem alfabética de seus sobrenomes, depois de seus nomes (vamos supor aqui que se dois candidatos tem o mesmo sobrenome, eles tem que ter nomes diferentes). Estes arquivos estão compactados no arquivo `candidatos.zip` do SIGAA.

Salve também o arquivo `testes.tgz` que contem os programas de teste que você vai utilizar nesta prática.

A classe `Candidato` é fornecida, e se encontra no arquivo `Candidato.h`. A classe contem o construtor que decompõe o string de entrada em um sobrenome, um nome e uma nota. Ela contem também um método `string toString()` que realiza a operação inversa, ou seja retorna uma string formada pelo sobrenome, o nome e a nota.

Aliás, todos os arquivos extraídos de `candidatos.tgz` descrevem as candidaturas das regiões contendo o nome da região na primeira linha, depois os diferentes candidatos um por linha com seu sobrenome, nome e nota. Por exemplo, o arquivo `candidatsPicardie.txt` contem:

```
Picardie
CATIGNY LEONTE 10
EDEINNE GEORGY 6
LANGRONNE LAURE 6
NELINCK KENZA 13
```

O arquivo `TesteCandidato.cpp` fornecido lê o arquivo `candidatsPicardie.txt` e cria os objetos da classe `Candidato`, fornecendo a saída a seguir:

```
nome da regiao: Picardie
criacao do candidato: CATIGNY LEONTE 10
criacao do candidato: EDEINNE GEORGY 6
criacao do candidato: LANGRONNE LAURE 6
criacao do candidato: NELINCK KENZA 13
```

2. Construindo a corrente de candidatos

Vamos armazenar todos os candidatos lidos, encadeando-os entre si, como em uma corrente de nós, cada nó permitindo acessar o seu conteúdo, do tipo `Candidato*`, e também o nó seguinte `next` da corrente.

Escreva uma classe `NoCandidato` com seus atributos e construtor de modo que o programa `TesteEx1.cpp` forneça a seguinte saida:

```
No comeco m vale: 0

Depois m aponta para o no FONFEC Sophie 13
e o seu no seguinte e 0

finalmente m referencia o no HADY Jacques 7
e tambem via m->next, o no FONFEC Sophie 13
que e o ultimo no, pois m->next->next vale 0
```

Note que a string "0" é associada à referência `NULL` em C++.

A figura 1 mostra o estado da memória e o conteúdo do ponteiro `m` depois de cada uma das atribuições do programa `TesteEx1.cpp`: uma corrente sem nós é representada pelo valor `NULL` e o campo `next` aponta para o `NoCandidato` seguinte na corrente.

Agora adicione na classe `NoCandidato` um método `string toString()` de modo que descomentando a última linha do programa `TesteEx1.cpp` obtem-se ao fim como saída:

```
HADY Jacques 7 -> FONFEC Sophie 13 -> 0
```

2. Encapsulamento de `NoCandidato` em `ListaCandidatos`

Uma vez que a lista vazia é representada por `NULL` é impossível ter métodos para a classe `NoCandidato` que representem todas as listas. Por exemplo, um método como `isEmpty()` que testa se a lista está vazia não pode ser escrito como um método de `NoCandidato`.

Para consertar este problema, basta encapsular `NoCandidato` dentro de uma outra classe, ou seja, definimos uma classe `ListaCandidatos` tendo um campo `head` do tipo `NoCandidato*` que aponta para o nó inicial da lista.

Implemente a classe `ListaCandidatos` contendo:

- um construtor `ListaCandidatos` que inicializa `head` com `NULL` (para criar uma lista vazia)
- um método `void adicioneComoHead(Candidato* c)` que cria um novo nó referenciando o objeto `Candidato c` e o adiciona à lista como novo head
- um método `bool estaVazia()` que retorna `true` se e somente se a lista estiver vazia

O programa `TesteEx2.cpp` deve produzir a seguinte saída:

```
Lista A vazia: 1
Lista B vazia: 0 FONFEC Sophie 13 -> 0
Lista C vazia: 0 HADY Jacques 7 -> FONFEC Sophie 13 -> 0
```

A figura 2 apresenta o estado da lista após cada chamada de método feita em `TesteEx2.cpp`

3. Número de elementos de `ListaCandidatos`

Implemente na classe `ListaCandidatos` um método `int tamamho()` que retorna o número de nós da lista a partir de seu nó inicial `head`. Você pode testar o seu código com o programa `TesteEx3a.cpp` que deve ter como saída:

```
Numero de nos da lista: 1
Numero de nos da lista: 3
```

Implemente também um método `string toString()` em `ListaCandidatos` que utiliza o método `toString()` da classe `NoCandidato` para retornar a cadeia contendo a descrição da lista. O programa `testeEx3b.cpp` deve produzir a saída a seguir:

```
lista de 0 candidatos: 0
lista de 1 candidatos: FONFEC Sophie 13 -> 0
lista de 2 candidatos: HADY Jacques 7 -> FONFEC Sophie 13 -> 0
```

4. Criando uma lista a partir de um arquivo

Inspirando-se do programa `TesteCandidato.cpp` adicione um construtor `ListaCandidatos(string nomeDoArquivo)` de modo que `TesteEx4.cpp` produza a saída a seguir:

```
criacao da lista de candidatos de: Bourgogne
lista de 5 candidatos: THOULIER SARAH 6 -> RABODOU CLEMENT 15 ->
CHESNEVARIN UGO 18 -> BEGIZ KENZA 9 -> ALLUIRE GERALDINE 15 -> 0
```

5. Remoção de nós

Pode ser que durante o processo de recrutamento um certo número de candidatos desista, por exemplo porque eles encontraram um emprego. Será necessário então retirá-los da lista de candidatos admissíveis. Em outros casos, a empresa vai querer eliminar os candidatos que não possuem uma nota suficientemente boa.

Escreva um método `bool remover(string nome, string sobrenome)` na classe `ListaCandidatos`. Se o candidato indicado por `nome` e `sobrenome` se encontra na lista, este método remove o nó correspondente e retorna o valor `true`, caso contrário o método não faz nada e retorna `false`. Se um candidato se encontrar mais de uma vez na lista, somente a primeira ocorrência é removida.

Este método vai necessitar da utilização de um ponteiro `NoCandidato* it` que vai se deslocar ao longo da lista. Preste atenção ao fato que a remoção de um nó se faz modificando-se o campo `next` do nó precedente, salvo no caso da remoção do nó `head`, que deve ser tratada de maneira especial. Além disso, se estiver programando em C++, é preciso liberar com o comando `delete` o espaço de memória ocupado pelo nó removido.

Para ajudá-lo, a figura 3 representa o estado da memória antes e depois da remoção do candidato `Sophie FONFEC` de uma lista de três elementos:

Você pode utilizar o método `bool igual(string sobrenome, string nome)` da classe `Candidato` para identificar um candidato a partir de seu nome e sobrenome.

Para testar esta função, utilize a classe `TesteEx5.cpp` e verifique se o resultado é este:

```
criacao da lista de candidatos de: Bourgogne
lista de 5 candidatos: THOULIER SARAH 6 -> RABODOU CLEMENT 15 ->
 CHESNEVARIN UGO 18 -> BEGIZ KENZA 9 -> ALLUIRE GERALDINE 15 -> 0

remocao feita; nova lista:
lista de 4 candidatos: THOULIER SARAH 6 -> RABODOU CLEMENT 15 ->
CHESNEVARIN UGO 18 -> ALLUIRE GERALDINE 15 -> 0

remocao feita; nova lista:
lista de 3 candidatos: THOULIER SARAH 6 -> RABODOU CLEMENT 15 -> ALLUIRE GERALDINE 15 ->

remocao feita; nova lista:
lista de 2 candidatos: RABODOU CLEMENT 15 -> ALLUIRE GERALDINE 15 -> 0

remocao nao realizada, LUC LEROI nao se encontra na lista
lista de 2 candidatos: RABODOU CLEMENT 15 -> ALLUIRE GERALDINE 15 -> 0

remocao feita; nova lista:
lista de 1 candidatos: ALLUIRE GERALDINE 15 -> 0

remocao feita; nova lista:
lista de 0 candidatos: 0

remocao nao realizada, CLEMENT RABODOU nao se encontra na lista
lista de 0 candidatos: 0
```

6. Filtragem de uma lista encadeada

O processo de recrutamente da empresa não se interessa por todos os candidatos, mais somente por aqueles com nota superior a um dado valor.

Implemente na classe `ListaCandidatos` um método `void filtrarCandidatos(int nota)` que conserva na lista apenas os candidatos com nota superior ou igual a `nota`. Este método

não deve criar novos objetos da classe `NoCandidato` e deve liberar o espaço dos nós removidos.

Para testar a função, execute o programa `TesteEx6.cpp`. Você deve obter o resultado a seguir:

```
criacao da lista de candidatos de: Centre
lista de 9 candidatos: VAUHEGHE ENOLA 14 -> SURTOUQUES GERALDY 9 ->
RASBET ENOLA 7 -> LANGRONNE ENZO 15 -> HYLEYN ANTOINE 13 -> HUISU GERALDE 12 ->
HENDLE LEONISE 6 -> FLOBARDO FABIO 10 -> BESREE LAURIE 9 -> 0
filtragem com nota = 13
lista de 3 candidatos: VAUHEGHE ENOLA 14 -> LANGRONNE ENZO 15 ->
HYLEYN ANTOINE 13 -> 0

criacao da lista de candidatos de: Bourgogne
lista de 5 candidatos: THOULIER SARAH 6 -> RABODOU CLEMENT 15 ->
CHESNEVARIN UGO 18 -> BEGIZ KENZA 9 -> ALLUIRE GERALDINE 15 -> 0
filtragem com nota = 20
lista de 0 candidatos: 0

criacao da lista de candidatos de: Limousin
lista de 4 candidatos: RAINEL ENOLA 8 -> MALAISEL CLEMENT 11 ->
GENOUIN CLARA 17 -> CENDRAY SARAH 17 -> 0
filtragem com nota = 0
lista de 4 candidatos: RAINEL ENOLA 8 -> MALAISEL CLEMENT 11 ->
GENOUIN CLARA 17 -> CENDRAY SARAH 17 -> 0
```

7. Concatenação de listas

Implemente o método `void concatena(ListaCandidatos* l)` que adiciona os nós da lista `l` ao fim da lista atual. Atenção! Este método não deve criar novos objetos da classe `NoCandidato`.

Teste o seu método com o programa `TesteEx7.cpp`. Você deve obter como saída do programa:

```
 criacao da lista de candidatos de: Bourgogne
lista de 5 candidatos: THOULIER SARAH 6 -> RABODOU CLEMENT 15 ->
CHESNEVARIN UGO 18 -> BEGIZ KENZA 9 -> ALLUIRE GERALDINE 15 -> 0

criacao da lista de candidatos de: Picardie
lista de 4 candidatos: NELINCK KENZA 13 -> LANGRONNE LAURE 6 ->
EDEINNE GEORGY 6 -> CATIGNY LEONTE 10 -> 0

concatenacao
```

```
lista de 9 candidatos: THOULIER SARAH 6 -> RABODOU CLEMENT 15 ->
CHESNEVARIN UGO 18 -> BEGIZ KENZA 9 -> ALLUIRE GERALDINE 15 ->
NELINCK KENZA 13 -> LANGRONNE LAURE 6 -> EDEINNE GEORGY 6 -> CATIGNY LEONTE 10 -> 0
```

*Este trabalho prático é de autoria de François Morain (Poly, France)

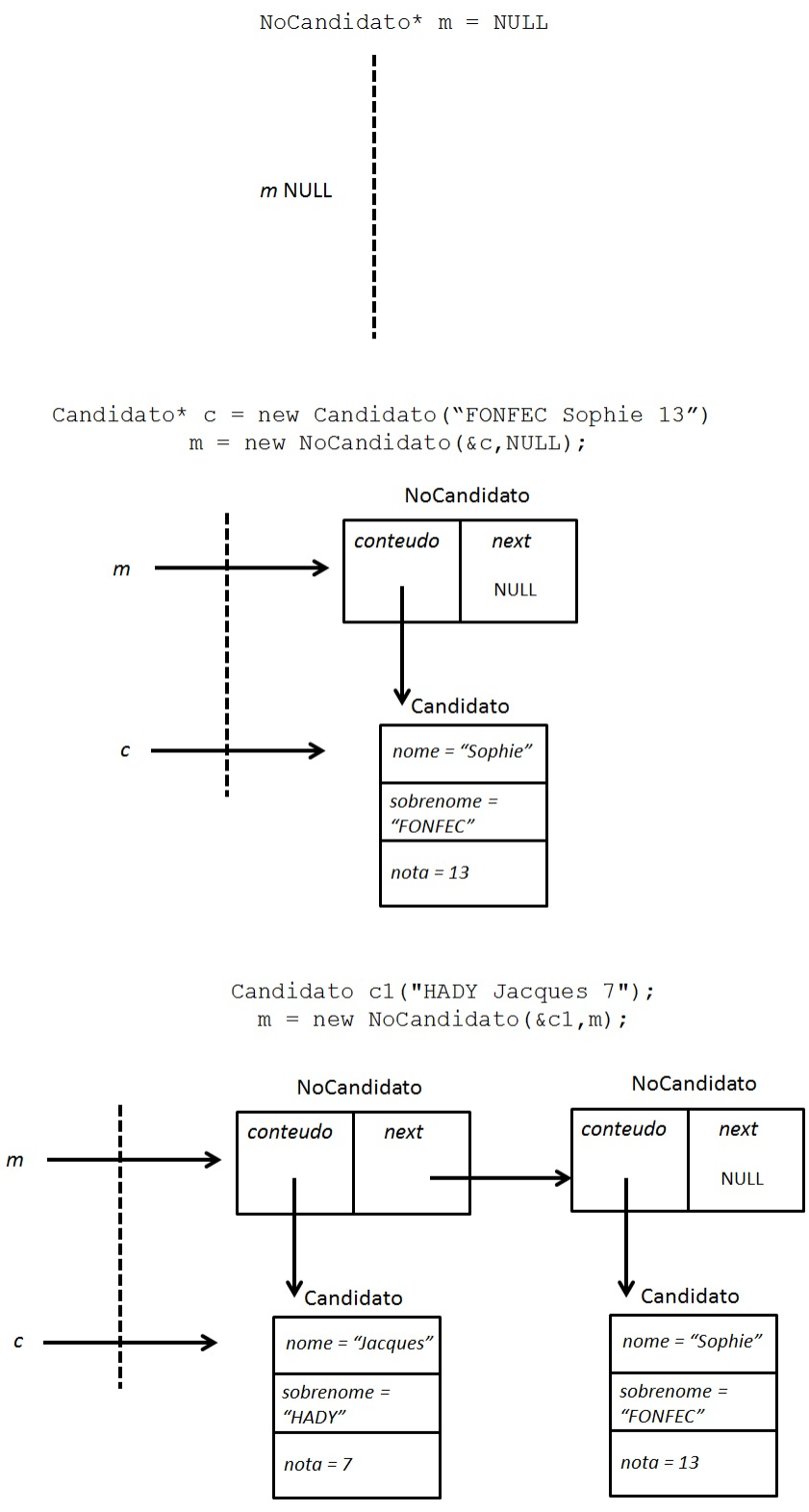

Figure 1: Exemplo do estado de memória em `TesteEx1.cpp`

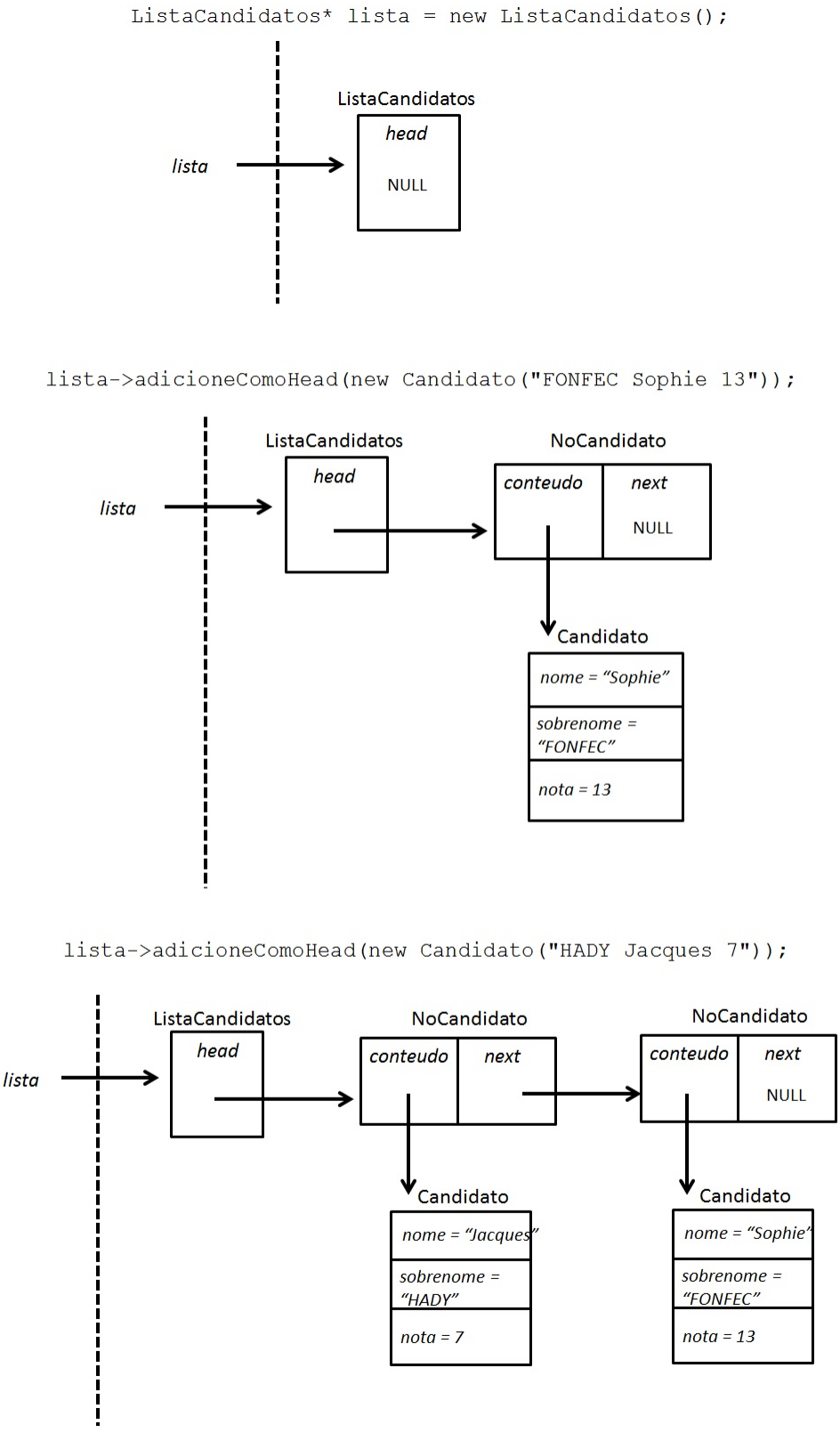

Figure 2: Exemplo do estado de memória em `TesteEx2.cpp`

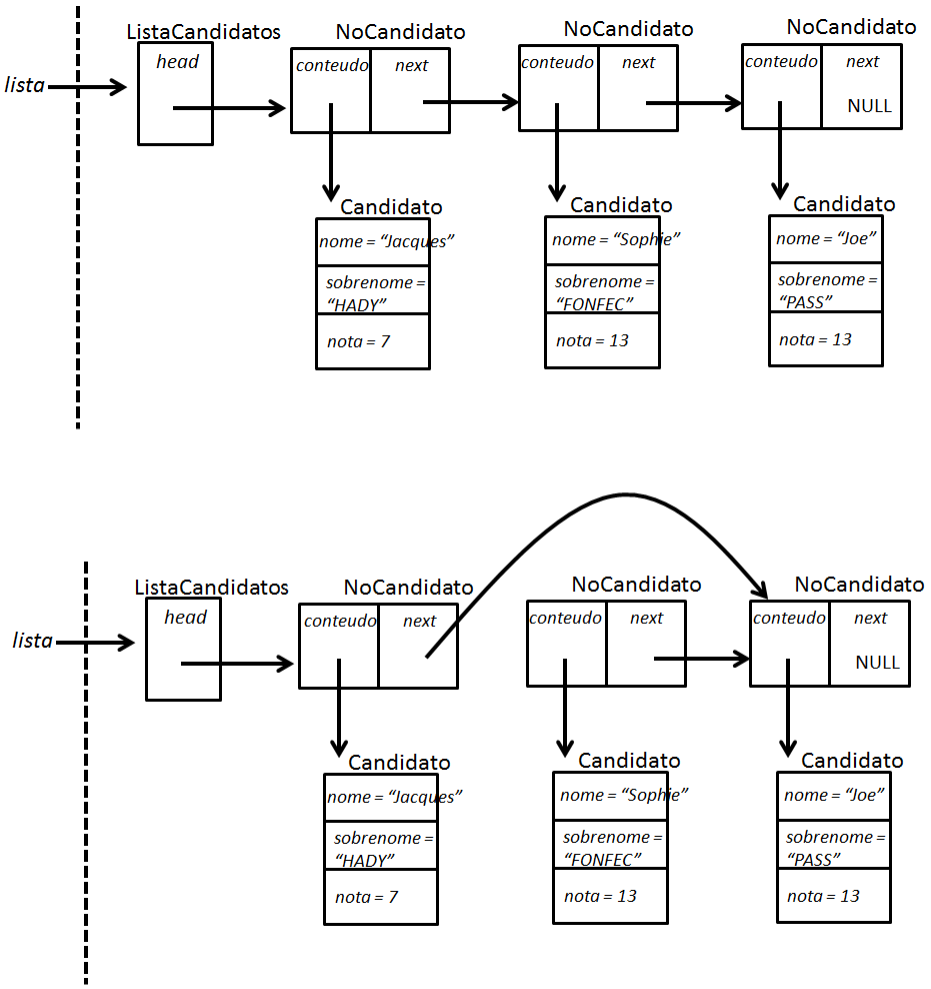

Figure 3: Exemplo de remoção