Teórico 10

Lucas Santiago de Oliveira

Maio de 2020

1 Agenda usando Árvore de lista

1.1 Agenda usada:

```
1  #include <iostream>
2  #include <string>
3  #include "Arvore.hpp"
4  #include <string.h>
5
6  #define bool short
7  #define true 1
8  #define false 0
9
10 //Criacao de uma Agenda em C++
11 class Agenda {
12 protected:
13     Arvore *arvore;
14
15 public:
16     //Contruindo uma Agenda
17     Agenda() {
18         arvore = new Arvore();
19     }
20
21     //Contruindo a Arvore da agenda com uma semente de criacao
22     Agenda(char seed[]) {
23         arvore = new Arvore(seed);
```

```
24      }
25
26      //Inserir um novo contato na Agenda
27      void inserir(string nome, string email, int telefone, int cpf)
        {
28          this->arvore->inserir(nome, email, telefone, cpf);
29      }
30
31      //Remover elemento da Agenda
32      void remover(string nome) {
33          this->arvore->remover(nome);
34      }
35
36      //Pesquisar elemento na Agenda pelo nome
37      bool pesquisar(string nome) {
38          return this->arvore->pesquisar(nome);
39      }
40
41      //Pesquisar elemento na Agenda pelo CPF
42      bool pesquisar(int cpf) {
43          return this->arvore->pesquisar(cpf);
44      }
45
46      //Mostrar contatos da Agenda
47      void mostrar() {
48          this->arvore->mostrar();
49      }
50  };
```

1.2 Árvore usada:

```
1  #ifndef Arvore_hpp
2  #define Arvore_hpp
3
4  #include <iostream>
5  #include "NoArvore.hpp"
6  #include "NoContato.hpp"
```

```
7  using namespace std;
8
9  class Arvore {
10 public:
11     NoArvore *raiz;
12
13     //Criacao de uma Arvore
14     Arvore() {
15         this->raiz = NULL;
16     }
17
18     //Criacao de uma Arvore usando seed
19     Arvore(char seed[]) {
20         this->raiz = NULL;
21         for(char i : seed) _seed(i, this->raiz);
22     }
23
24     //Mostrar elementos da Arvore
25     void mostrar() {
26         this->_mostrar(this->raiz);
27         cout << "----------------------------" << endl;
28     }
29
30     //Inserir contato na Arvore
31     void inserir(string nome, string email, int telefone, int cpf)
       {
32         NoContato *noContato = new NoContato(nome, email, telefone,
        cpf);
33
34         if(this->raiz == NULL) {
35             this->raiz = new NoArvore();
36             this->raiz->inserir(noContato);
37
38         } else _inserir(noContato, this->raiz);
39
40     }
41
42     //Remover um elemento da Agenda a partir do nome
```

```
43      void remover(string nome) {
44          _remover(this->raiz, nome);
45
46      }
47
48      //Pesquisar um elemento na Arvore usando o nome
49      bool pesquisar(string nome) {
50          return _pesquisar(this->raiz, nome);
51      }
52
53      //Pesquisar um elemento na Arvore usando o CPF
54      bool pesquisar(int cpf) {
55          return _pesquisar(this->raiz, cpf);
56      }
57
58  private:
59      //Criacao de uma arvore a partir de uma seed
60      void _seed(char semente, NoArvore *i) {
61          if(this->raiz == NULL) {
62              this->raiz = new NoArvore();
63              this->raiz->primeiraLetraNome = semente;
64
65          } else {
66              if(i->primeiraLetraNome < semente) {
67                  if(i->esq == NULL) i->esq = new NoArvore(semente);
68                  else _seed(semente, i->esq);
69
70              } else if(i->primeiraLetraNome > semente) {
71                  if(i->dir == NULL) i->dir = new NoArvore(semente);
72                  else _seed(semente, i->dir);
73
74              }
75          }
76      }
77
78      //Percorrer a arvore para inserir um contato
79      void _inserir(NoContato *inserir, NoArvore *i) {
80          if(inserir->contato->nome[0] < i->primeiraLetraNome) {
```

```
81              if(i->esq == NULL) i->esq = new NoArvore(inserir);
82              else _inserir(inserir, i->esq);
83
84          } else if(inserir->contato->nome[0] > i->primeiraLetraNome)
        {
85              if(i->dir == NULL) i->dir = new NoArvore(inserir);
86              else _inserir(inserir, i->dir);
87
88          } else cerr << "Impossivel inserir "
89                      << inserir->contato->nome
90                      << " nome repetido!" << endl;
91      }
92
93      //Percorrer a Arvore e remover um elemento pelo nome
94      void _remover(NoArvore* i, string nome) {
95          if(i == NULL) cerr << "Elemento nao encontrado!" << endl;
96          else if(nome[0] < i->primeiraLetraNome) _remover(i->esq,
        nome);
97          else if(nome[0] > i->primeiraLetraNome) _remover(i->dir,
        nome);
98          else i->remover(nome);
99
100     }
101
102     //Percorrer elemento na Arvore pesquisando por nome
103     bool _pesquisar(NoArvore* i, string nome) {
104         bool resp = false;
105         if(i != NULL && nome[0] < i->primeiraLetraNome) resp =
        _pesquisar(i->esq, nome);
106         else if(nome[0] > i->primeiraLetraNome) resp = _pesquisar(i
        ->esq, nome);
107         else {
108             resp = i->pesquisar(nome);
109             return resp;
110
111         }
112     }
113
```

```
114     //Percorrer arvore escrevendo valor contido nela
115     void _mostrar(NoArvore *i) {
116         if(i != NULL) {
117            _mostrar(i->esq);
118            cout << "----------------------------" << endl;
119            i->mostrar();
120            _mostrar(i->dir);
121         }
122     }
123 };
124 
125 #endif
```

1.3 No da árvore usada:

```
1  #ifndef NoArvore_hpp
2  #define NoArvore_hpp
3  
4  #include <iostream>
5  #include "NoContato.hpp"
6  using namespace std;
7  
8  class NoArvore {
9  public:
10     char primeiraLetraNome;
11     NoContato *inicio, *fim;
12     NoArvore *esq, *dir;
13 
14     //Inicializando um No da Arvore da Agenda
15     NoArvore() {
16         this->esq = this->dir = NULL;
17         this->inicio = this->fim = NULL;
18     }
19 
20     //Criando um no de Arvore com seed
21     NoArvore(char seed) {
22         this->esq = this->dir = NULL;
```

```
23          this->inicio = this->fim = NULL;
24          this->primeiraLetraNome = seed;
25      }
26
27      //Inserir novos elementos na fila de contatos
28      void inserir(NoContato *inserir) {
29          if(this->inicio == NULL) {
30              this->inicio = this->fim = inserir;
31              this->primeiraLetraNome = inserir->contato->nome[0];
32
33          } else {
34              this->fim->prox = inserir;
35              this->fim = this->fim->prox;
36
37          }
38      }
39
40      //Pesquisar elemento pelo nome
41      bool pesquisar(string nome) {
42          return _pesquisar(this->inicio, nome);
43      }
44
45      //Remover elemento da Agenda
46      void remover(string nome) {
47          _remover(this->inicio, nome);
48      }
49
50      //Mostrar elementos presentes na lista de contatos
51      void mostrar() {
52          _mostrar(this->inicio);
53      }
54
55  private:
56      //Remover elemento da lista
57      void _remover(NoContato *i, string nome) {
58          if(i == NULL) cerr << "Elemento nao encontrado!" << endl;
59          else if(i->prox->contato->nome.compare(nome) == 0) {
60              NoContato *hold = i->prox->prox;
```

```
61              free(i->prox);
62              i->prox = hold;
63
64          }
65      }
66
67      //Percorrer a Lista procurando um elemento pelo nome
68      bool _pesquisar(NoContato *i, string nome) {
69          bool resp = false;
70
71          if(i != NULL) resp = _pesquisar(i->prox, nome);
72          else if(i->contato->nome.compare(nome) == 0) resp = true;
73
74          return resp;
75      }
76
77      //Percorrer todos os contatos da lista mostrando na tela
78      void _mostrar(NoContato *i) {
79          if(i != NULL) {
80              i->contato->mostrar();
81              _mostrar(i->prox);
82          }
83      }
84  };
85
86  #endif
```

1.4 Contato usada:

```
1  #ifndef Contato_hpp
2  #define Contato_hpp
3
4  #include <iostream>
5  #include <string>
6
7  using namespace std;
8
```

```
9  class Contato {
10 public:
11     string nome;
12     string email;
13     int telefone;
14     int cpf;
15
16     //Criando um Contato
17     Contato() {
18         this->nome      = "";
19         this->email     = "";
20         this->telefone  = 0;
21         this->cpf       = 0;
22     }
23
24     //Inserir um novo contato
25     void inserir(string nome, string email, int telefone, int cpf)
       {
26         this->nome       = nome;
27         this->email      = email;
28         this->telefone   = telefone;
29         this->cpf        = cpf;
30     }
31
32     void mostrar() {
33         cout << "Nome   - " << this->nome     << "\n"
34                 "Email  - " << this->email    << "\n"
35                 "Numero - " << this->telefone << "\n"
36                 "CPF    - " << this->cpf      << endl;
37     }
38 };
39
40 #endif
```

1.5 Celula de contato usada:

```
1 #ifndef NoContato_hpp
```

```
#define NoContato_hpp

#include <iostream>
#include <string>
#include "Contato.hpp"

using namespace std;

class NoContato {
public:
    Contato *contato;
    NoContato *prox;

    //Criacao de um No
    NoContato() {
        this->prox = NULL;
    }

    //Criacao de um no com um valor
    NoContato(string nome, string email, int telefone, int cpf) {
        this->contato->inserir(nome, email, telefone, cpf);
        this->prox = NULL;
    }

    //Criar um novo contato
    void inserir(Contato *inserir) {
        contato = inserir;
    }
};
#endif
```