Teórico IX

Lucas Santiago de Oliveira

Maio de 2020

1 Questão 1

```
1  /**
2  * Metodo para retornar elemento removido de uma arvore
3  * @param x valor para ser removido da arvore
4  * @return Elemento presente na arvore
5  */
6  public int remover3(int x) throws Exception {
7    int saida = -1;
8    Celula tmp = this.raiz;
9
10   if(tmp == null) throw new Exception("Erro ao remover de arvore
       vazia!");
11   else if(x < tmp.elemento) saida = _remover3(x, tmp.esq, tmp);
12   else if(x > tmp.elemento) saida = _remover3(x, tmp.dir, tmp);
13   else if(tmp.dir == null) {
14     saida = tmp.elemento;
15     this.raiz = this.raiz.esq;
16
17   } else if(tmp.esq == null) {
18     saida = tmp.elemento;
19     this.raiz = this.raiz.dir;
20
21   } else {
22     saida = tmp.elemento;
23     this.raiz.esq = antecessor(tmp, tmp.esq);
24
```

```
25    }
26    return saida;
27  }
28
29  /**
30  * @param x Elemento para ser removido da arvore
31  * @param i No que estao sendo pesquisado no momento
32  * @param pai No pai do No i
33  * @return Elemento a ser removido
34  */
35  private int _remover3(int x, No i, No pai) throws Exception {
36    int saida = 0;
37
38    if(i == null) throw new Exception("Elemento nao encontrado na
        arvore!");
39    else if(x < i.elemento) saida = _remover3(x, i.esq, i);
40    else if(x > i.elemento) saida = _remover3(x, i.dir, i);
41    else if(i.dir == null) {
42      saida = i.elemento;
43
44      if(pai.esq == i) {
45        pai.esq = i.esq;
46
47      } else {
48        pai.dir = i.esq;
49
50      }
51    } else if(i.esq == null) {
52      saida = i.elemento;
53
54      if(pai.esq == i) {
55        pai.esq = i.dir;
56
57      } else {
58        pai.dir = i.dir;
59
60      }
61    } else {
```

```
62      resp = i.elemento;
63      i.esq = antecessor(i, i.esq);
64
65    }
66
67    return saida;
68  }
```

2 Questão 2

```
1  #include <iostream>
2  #include <err.h>
3
4  //Criacao da classe No em C++
5  class No {
6  public:
7    No *esq, *dir;
8    int elemento;
9
10   //Contrutor da classe vazia de No
11   No() {
12     this->elemento = 0;
13   }
14
15   //Criacao de um No
16   No(int elemento) {
17     this->elemento = elemento;
18     this->esq = this->dir = NULL;
19   }
20 };
21
22 //Criacao do tipo Arvore binaria em C++
23 class Arvore {
24 public:
25   No *raiz;
26
```

```
27    //Criando a arvore
28    Arvore() {
29      this->raiz = NULL;
30    }
31
32    //Inserir elemento na Arvore
33    void inserir(int x) {
34      if(this->raiz == NULL) this->raiz = new No(x);
35      else _inserir(this->raiz, x);
36
37    }
38
39    //Implementacao do treeSort em C++
40    void treeSort() {
41      std::cout << "[ ";
42      _treeSort(this->raiz);
43      std::cout << "]\n";
44    }
45
46  private:
47    //Percorrer a arvore para inserir um elemento
48    void _inserir(No *no, int x) {
49      if(x < no->elemento) {
50        if(no->esq == NULL) no->esq = new No(x);
51        else _inserir(no->esq, x);
52
53      } else if(x > no->elemento) {
54        if(no->dir == NULL) no->dir = new No(x);
55        else _inserir(no->dir, x);
56
57      } else warn("O item j   est   contido na Arvore, por isso, nao
        foi inserido!");
58    }
59
60    //Percorrendo a arvore e escrevendo ordenado
61    void _treeSort(No* no) {
62      if(no != NULL) {
63        _treeSort(no->esq);
```

```
64        std::cout << no->elemento << " ";
65        _treeSort(no->dir);
66      }
67    }
68 };
69
70 int main(int argc, char **argv) {
71    Arvore arvore;
72    int entrada;
73
74    //Inserir elementos na arvore para serem ordenados
75    std::cout << "Digite 0 para terminar o programa!\n";
76    std::cin >> entrada;
77    while(entrada != 0) {
78      arvore.inserir(entrada);
79      std::cin >> entrada;
80    }
81
82    arvore.treeSort();
83 }
```