ESTRUTURAS DE DADOS

Árvores (Continuação)

Roteiro

- Detalhes de Implementação
  - Remoção de um nó
- Aplicações da Estrutura

Detalhes de Implementação

Iremos efetuar agora a remoção de um nó da árvore. O nosso código é dividido em algumas etapas:

- **deleteAluno: um método que navegará pela arvore até encontrar o nó a ser excluído.**
- **deleteNode: um método que receberá por parâmetro o nó a ser excluído e tratará três casos:**
  1. **O nó é uma folha.**
  2. **O nó tem um filho.**
  3. **O nó tem dois filhos (busca pelo sucessor).**

- **Removendo aluno: localizando o aluno a ser removido**
  - **Não muda em nada o conceito de uma busca convencional.**

```
void SearchTree::deleteAluno(NodeType*& tree, int aluno)
{
  if (aluno < tree->aluno.getRa() )
    deleteAluno(tree->esquerda, aluno);
  else if (aluno > tree->aluno.getRa() )
    deleteAluno(tree->direita, aluno);
  else if (aluno == tree->aluno.getRa())
    deleteNode(tree);
}
```

- **Removendo aluno: o nó a ser removido não tem dois filhos.**
  - **Na pior das hipóteses, substituímos o pai pelo filho e removemos.**

```
void SearchTree::deleteNode(Node*& tree)
{
  Aluno data;
  Node* tempPtr;
  tempPtr = tree;
  if (tree->esquerda == NULL)
    {
      tree = tree->direita;
      delete tempPtr;
    }
  else if (tree->direita == NULL)
    {
      tree = tree->esquerda;
      delete tempPtr;
    }
```

Nesses casos, pelo menos um dos filhos é NULL.

- **Removendo aluno: o próximo caso precisará do predecessor lógico ou do sucessor lógico.**
  - **Escolhemos utilizar o sucessor lógico:**

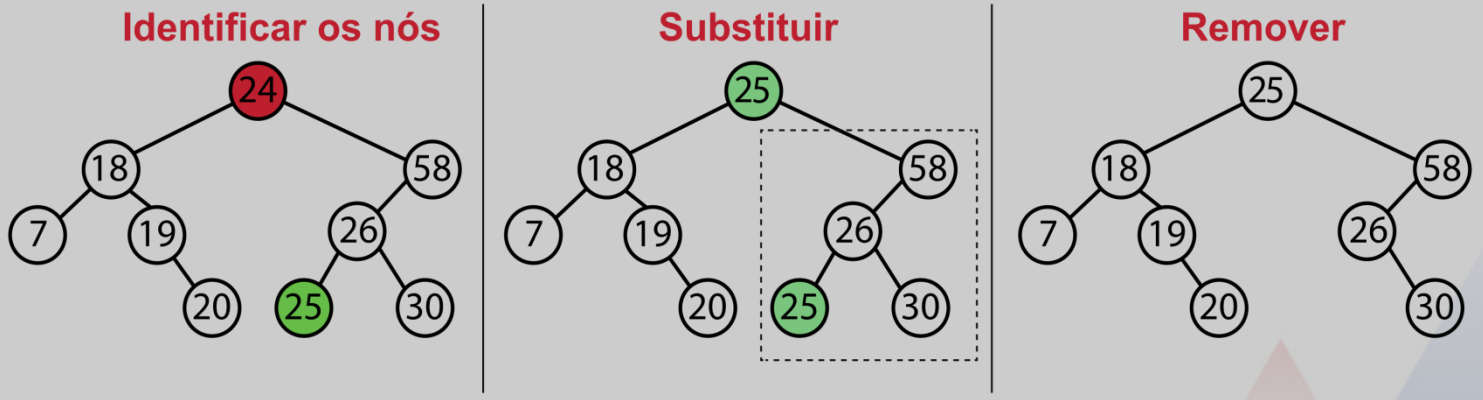
```
void SearchTree::getSuccessor(NodeType* tree, Aluno& data)
{
  tree = tree->direita;
  while (tree->esquerda != NULL)
    tree = tree->esquerda;
  data = tree->aluno;
}
```

  - **O sucessor lógico é o filho mais à esquerda da árvore da direita.**
  - **Procuramos esse nó e retornamos no parâmetro data, passado por referência.**

- **Removendo aluno: podemos substituir o sucessor com o nó a ser removido.**

```
getSuccessor(tree, data);
tree->aluno = data;
deleteAluno(tree->direita, data.getRa());
```

  - **Observe que isso fará com que momentaneamente a árvore tenha dois nós iguais.**
  - **Invocamos deleteAluno recursivamente para apagar o nó duplicado na árvore da direita.**
  - **O algoritmo entrará em looping infinito?**

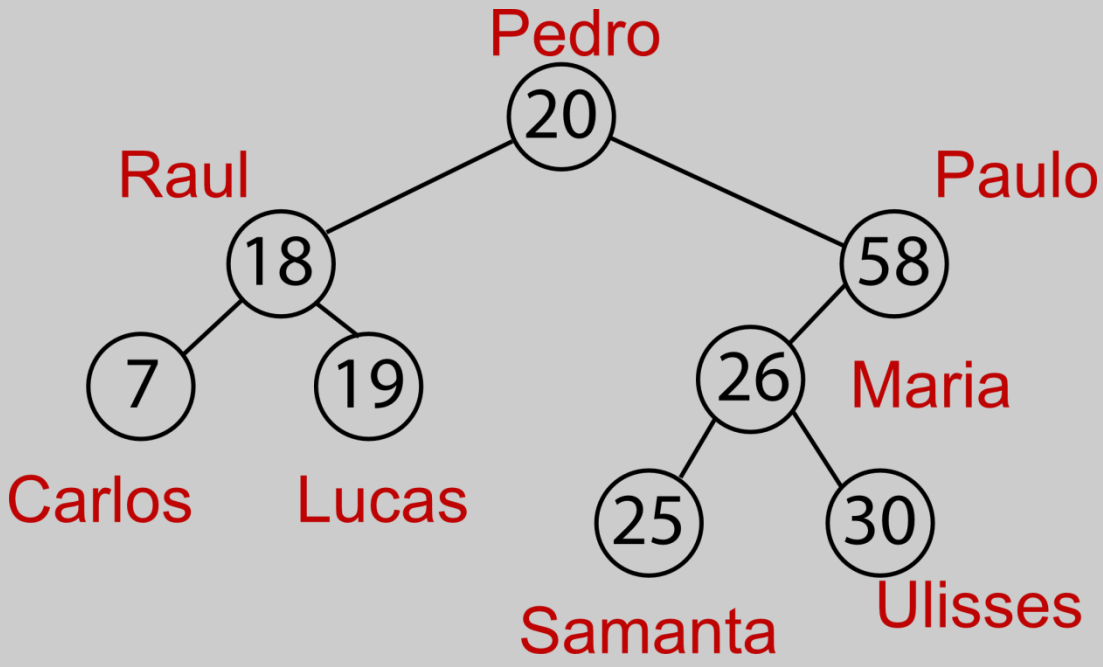


```
void SearchTree::deleteNode(NodeType*& tree) {
  Aluno data;
  NodeType* tempPtr;
  tempPtr = tree;
  if (tree->esquerda == NULL) {
    tree = tree->direita;
    delete tempPtr;
  }
  else if (tree->direita == NULL) {
    tree = tree->esquerda;
    delete tempPtr;
  }
  else {
    getSuccessor(tree, data);
    tree->aluno = data;
    deleteAluno(tree->direita, data);
  }
}
```

Código completo

Aplicações da Estrutura

Vamos agora inserir alguns elementos e efetuar algumas operações.

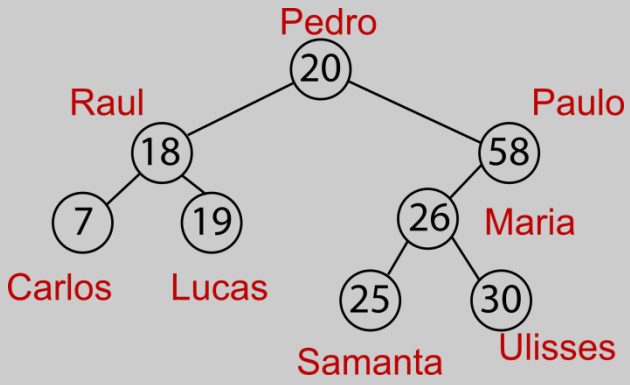


Inserindo os dados:

```
const int NUM_ALUNOS = 8;

int main() {
  SearchTree searchTree;

  int ras[NUM_ALUNOS] = {20, 18, 58, 7, 19, 26, 25, 30};
  string nomes[NUM_ALUNOS] = {
    "Pedro", "Raul", "Paulo",
    "Carlos", "Lucas", "Maria",
    "Samanta", "ulisses"};
  Aluno alunos[NUM_ALUNOS];

  for (int i = 0; i < NUM_ALUNOS; i++){
    Aluno aluno = Aluno(ras[i], nomes[i]);
    alunos[i] = aluno;
    searchTree.insertAluno(aluno);
  }
```

Imprimindo na saída padrão

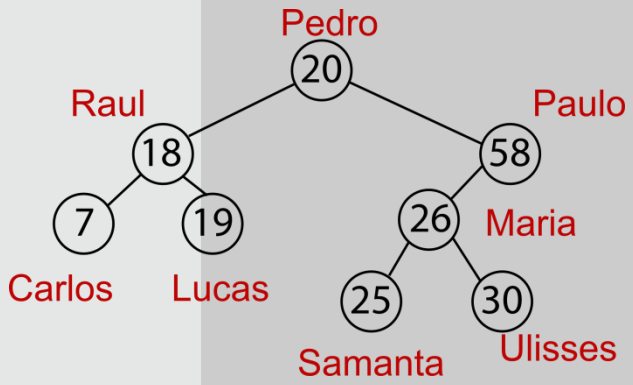
```
cout << "Pre:  ";
searchTree.printPreOrder();
cout << endl;
cout << "In:   ";
searchTree.printInOrder();
cout << endl;
cout << "Post: ";
searchTree.printPostOrder();
cout << endl;
```




```
Pre:  Pedro , Raul , Carlos , Lucas , Paulo , Maria , Samanta , ulisses ,
In:   Carlos , Raul , Lucas , Pedro , Samanta , Maria , ulisses , Paulo ,
Post: Carlos , Lucas , Raul , Samanta , ulisses , Maria , Paulo , Pedro ,
```

Removendo Pedro e imprimindo novamente

```
// Removendo aluno na raiz;
searchTree.deleteAluno(alunos[0].getRa());
cout << "********" << endl;
cout << "Pre:  ";
searchTree.printPreOrder();
cout << endl;
cout << "In:   ";
searchTree.printInOrder();
cout << endl;
cout << "Post: ";
searchTree.printPostOrder();
cout << endl;
```




```
Pre:  Samanta , Raul , Carlos , Lucas , Paulo , Maria , ulisses ,
In:   Carlos , Raul , Lucas , Samanta , Maria , ulisses , Paulo ,
Post: Carlos , Lucas , Raul , ulisses , Maria , Paulo , Samanta ,
```

ESTRUTURAS DE DADOS

Árvores (Continuação)