AULA e08
Algoritmos e Estruturas de Dados I

Árvores Binárias de Pesquisa - Inserção e Exclusão

(outras implementações)

Luciano Antonio Digiampietri

Árvores Binárias de Pesquisa

Aprenderemos implementações alternativas para as as funções de inserção e exclusão

Árvores Binárias de Pesquisa

Aprenderemos implementações alternativas para as as funções de inserção e exclusão

- Elas receberão como parâmetro um referência a variável que guarda o endereço de um nó;

Árvores Binárias de Pesquisa

Aprenderemos implementações alternativas para as as funções de inserção e exclusão

- Elas receberão como parâmetro um referência a variável que guarda o endereço de um nó;

- Serão utilizadas como base para os respectivos códigos de árvores AVL.

Inserção

A função receberá: endereço da variável que guarda a referência ao nó raiz e a chave a ser inserida.

Seguiremos as regras de inserção em árvores binárias de pesquisa:

Inserção

Seguiremos as regras de inserção em árvores binárias de pesquisa:

- Se a referência atual aponta para uma variável com valor nulo: realizamos a inserção;

Inserção

Seguiremos as regras de inserção em árvores binárias de pesquisa:

- Se a referência atual aponta para uma variável com valor nulo: realizamos a inserção;
- Caso contrário, seguiremos, recursivamente, para a esquerda ou para a direita de acordo com o valor da chave (valores maiores são inseridos à direita e menores ou iguais à esquerda).

Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){




}
```

Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }




}
```

Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);

  }
}
```

Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}
```

Inserção

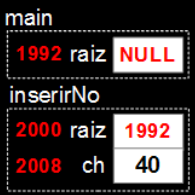
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
```

Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
```

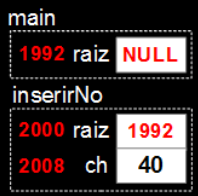
Inserção

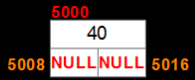
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
```



Inserção

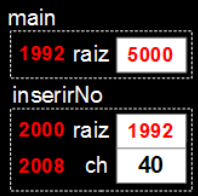
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
```

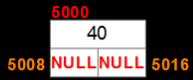




Inserção

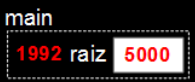
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
```

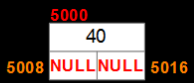




Inserção

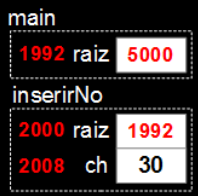
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
```

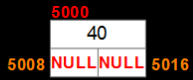




Inserção

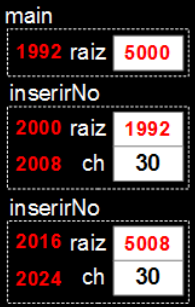
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
```

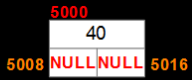




Inserção

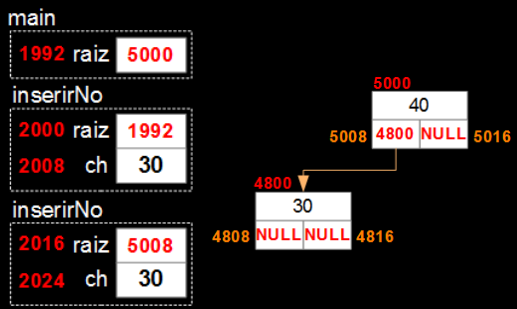
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
```





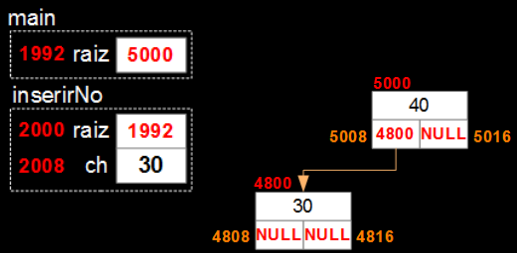
Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
```



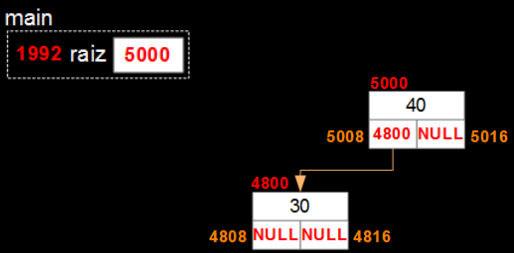
Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
```



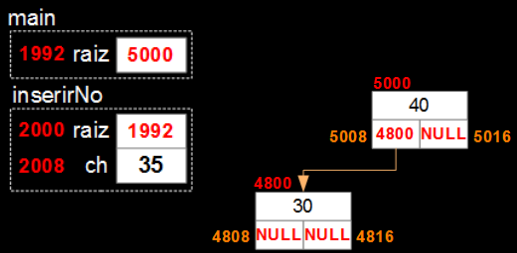
Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
```



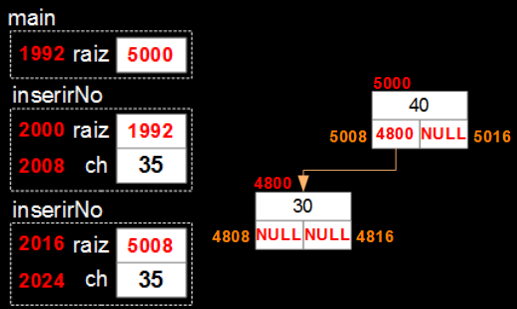
Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
  inserirNo(&raiz, 35);
```



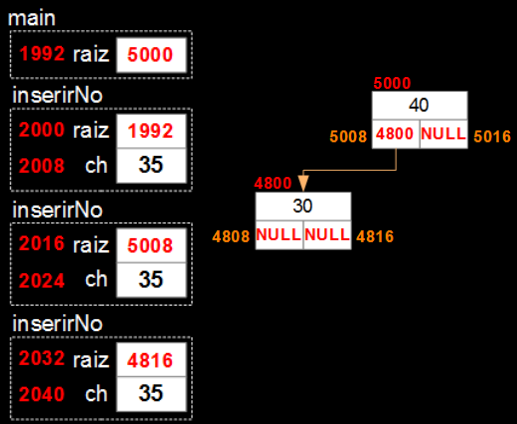
Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
  inserirNo(&raiz, 35);
```



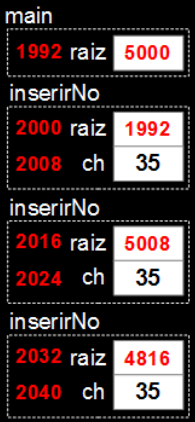
Inserção

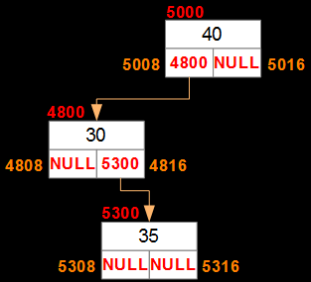
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
  inserirNo(&raiz, 35);
```



Inserção

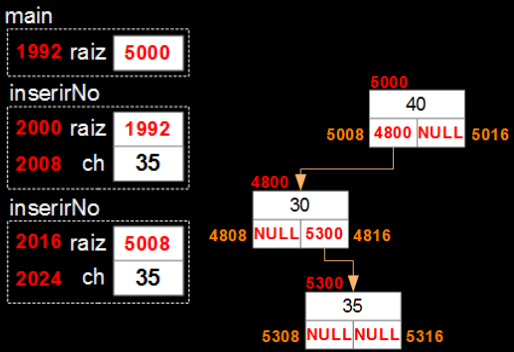
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
  inserirNo(&raiz, 35);
```





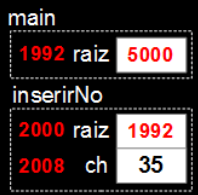
Inserção

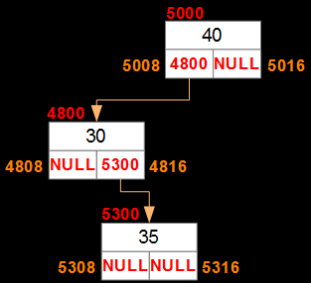
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
  inserirNo(&raiz, 35);
```



Inserção

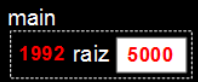
```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
  inserirNo(&raiz, 35);
```

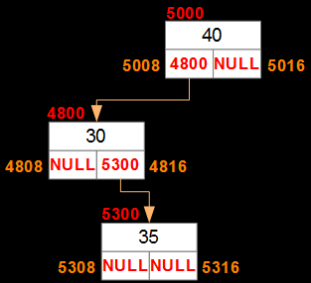




Inserção

```
bool inserirNo(PONT* raiz, TIPOCHAVE ch){
  if (*raiz == NULL){
    *raiz = criarNovoNo(ch);
    return true;
  }else{
    if ((*raiz)->chave >= ch)
       return inserirNo(&(*raiz)->esq, ch);
    else return inserirNo(&(*raiz)->dir, ch);
  }
}

int main(){
  PONT raiz = NULL;
  inserirNo(&raiz, 40);
  inserirNo(&raiz, 30);
  inserirNo(&raiz, 35);
```





Exclusão

A função receberá: endereço da variável que guarda a referência ao nó raiz e a chave do elemento a ser excluído.

Seguiremos as regras de exclusão em árvores binárias de pesquisa:

Exclusão

- **Buscaremos o elemento a ser excluído (se não existir, retornaremos *false*), mas esta busca será recursiva dentro da própria função;**

Exclusão

- **Buscaremos o elemento a ser excluído** (se não existir, retornaremos *false*), mas esta busca será **recursiva dentro da própria função**;
- Caso o elemento exista há **três casos**: **não possui filhos**, **possui um filho** (estes dois casos serão tratados juntos), e **possui dois filhos**.

Exclusão

- **Buscaremos o elemento a ser excluído** (se não existir, retornaremos *false*), mas esta busca será **recursiva dentro da própria função**;
- Caso o elemento exista há **três casos**: **não possui filhos**, **possui um filho** (estes dois casos serão tratados juntos), e **possui dois filhos**.
- Para o caso de possuir **dois filhos**, realizaremos a **substituição pelo *maior elemento à esquerda***.

Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;









}
```

Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;









}
```

Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;









  }


}
```

Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }




  }


}
```

Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }


}
```

Inserção

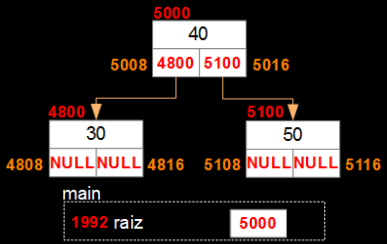
```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);

}
```

Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);
  else return excluirNo(&(atual->esq), ch);
}
```

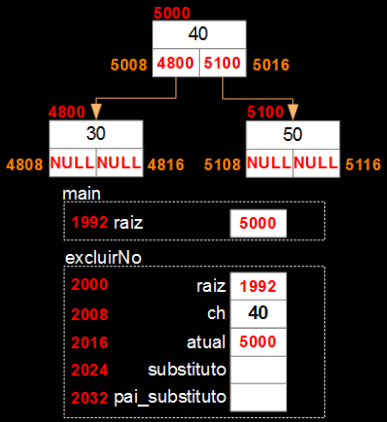
Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);
  else return excluirNo(&(atual->esq), ch);
}
```



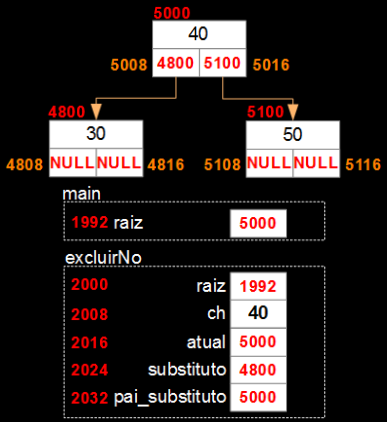
Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);
  else return excluirNo(&(atual->esq), ch);
}
```



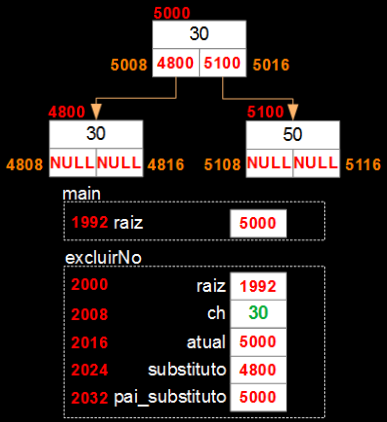
Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);
  else return excluirNo(&(atual->esq), ch);
}
```



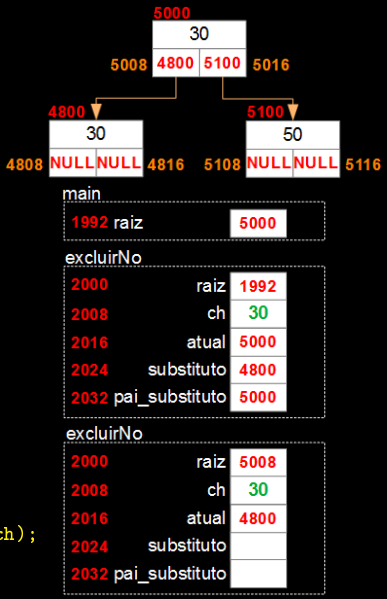
Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);
  else return excluirNo(&(atual->esq), ch);
}
```



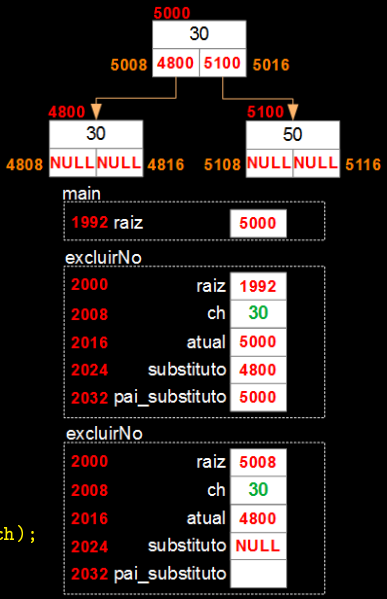
Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);
  else return excluirNo(&(atual->esq), ch);
}
```



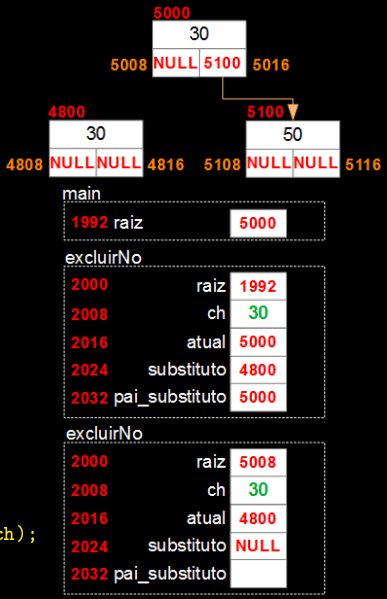
Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);
  else return excluirNo(&(atual->esq), ch);
}
```



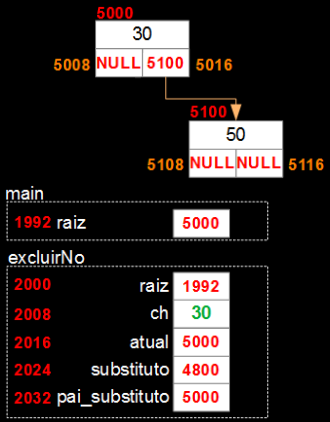
Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);
  else return excluirNo(&(atual->esq), ch);
}
```



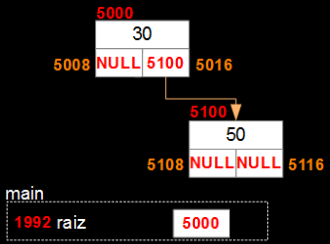
Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);
  else return excluirNo(&(atual->esq), ch);
}
```



Inserção

```
bool excluirNo(PONT* raiz, TIPOCHAVE ch){
  PONT atual = *raiz;
  if (!atual) return false;
  if (atual->chave == ch){
    PONT substituto, pai_substituto;
    if(!atual->esq || !atual->dir) {
      if(atual->esq) substituto = atual->esq;
      else substituto = atual->dir;
      *raiz = substituto;
      free(atual);
      return true;
    }else {
      substituto = maiorAEsquerda(atual,&pai_substituto);
      atual->chave = substituto->chave;
      ch = substituto->chave;
    }
  }
  if (ch > atual->chave) return excluirNo(&(atual->dir), ch);
  else return excluirNo(&(atual->esq), ch);
}
```



AULA e08
Algoritmos e Estruturas de Dados I

Árvores Binárias de Pesquisa - Inserção e Exclusão

(outras implementações)

Luciano Antonio Digiampietri