AULA 17
ESTRUTURA DE DADOS

Árvores Binárias de Pesquisa

Norton T. Roman & Luciano A. Digiampietri

Árvores Binárias de Pesquisa

Para nossa árvore binária de pesquisa, veremos as seguintes funções:

Inicialização da árvore

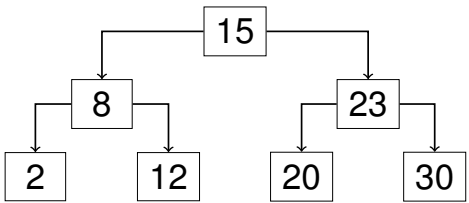
Inserção de um elemento

Busca de um elemento

Contagem do número de elementos

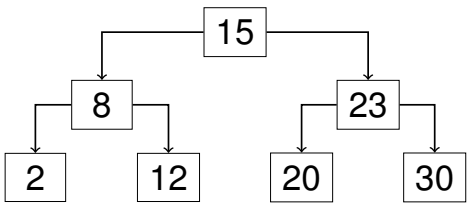
Leitura da árvore

Remoção de um elemento



ABP – Busca

Em uma Árvore de Busca Binária, agimos como na busca binária em um arranjo. Ex: *contem(12)*

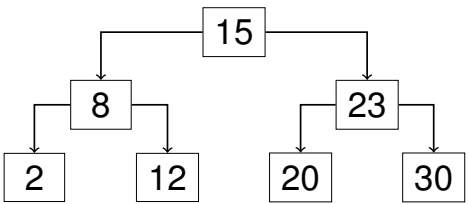


ABP – Busca

Em uma Árvore de Busca Binária, agimos como na busca binária em um arranjo. Ex: *contem(12)*

Primeiro olhamos a raiz:

- Se for 12, achamos
- Se for $>$ 12 olhamos na subárvore da esquerda
- Senão olhamos na da direita

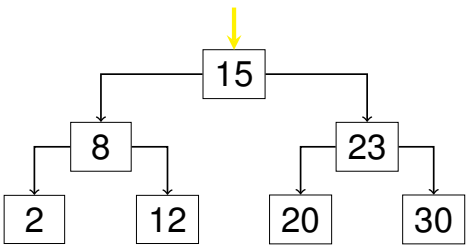


ABP – Busca

Em uma Árvore de Busca Binária, agimos como na busca binária em um arranjo. Ex: *contem(12)*

Primeiro olhamos a raiz:

- Se for 12, achamos
- Se for $>$ 12 olhamos na subárvore da esquerda
- Senão olhamos na da direita

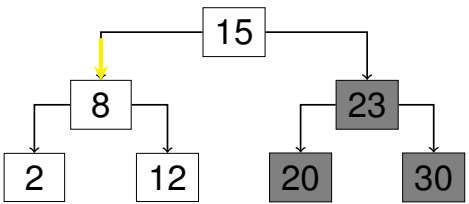


ABP – Busca

Em uma Árvore de Busca Binária, agimos como na busca binária em um arranjo. Ex: *contem(12)*

Primeiro olhamos a raiz:

- Se for 12, achamos
- Se for $>$ 12 olhamos na subárvore da esquerda
- Senão olhamos na da direita

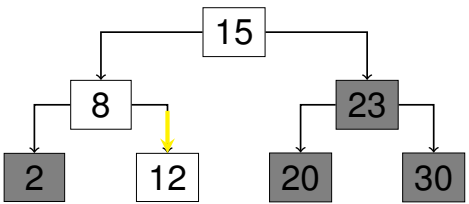


ABP – Busca

Em uma Árvore de Busca Binária, agimos como na busca binária em um arranjo. Ex: *contem(12)*

Primeiro olhamos a raiz:

- Se for 12, achamos
- Se for $>$ 12 olhamos na subárvore da esquerda
- Senão olhamos na da direita



ABP – Busca

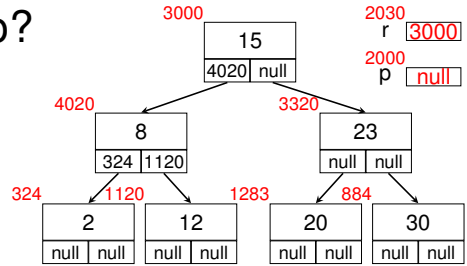
E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

ABP – Busca

E como fica o código para isso?

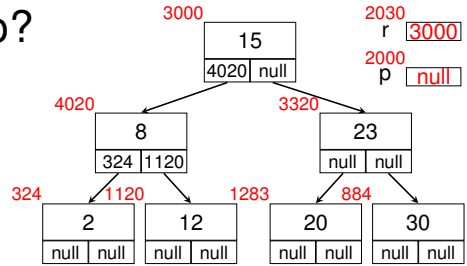
```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```



ABP – Busca

E como fica o código para isso?

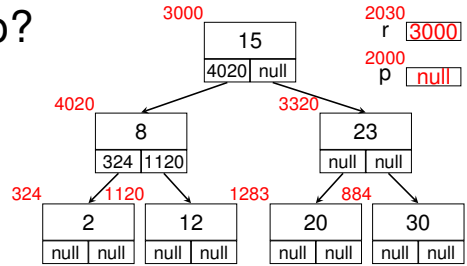
```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```



ABP – Busca

E como fica o código para isso?

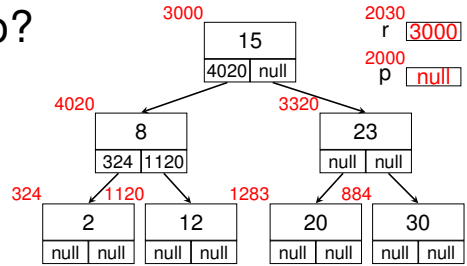
```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```



ABP – Busca

E como fica o código para isso?

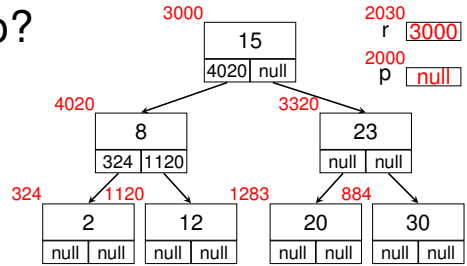
```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```



ABP – Busca

E como fica o código para isso?

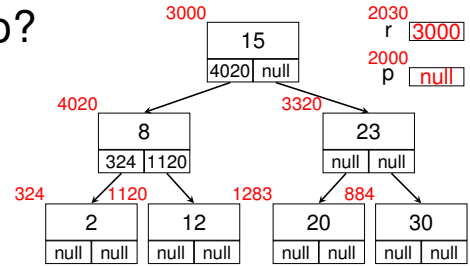
```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

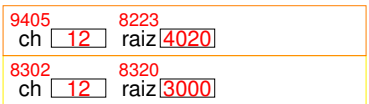


ABP – Busca

E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

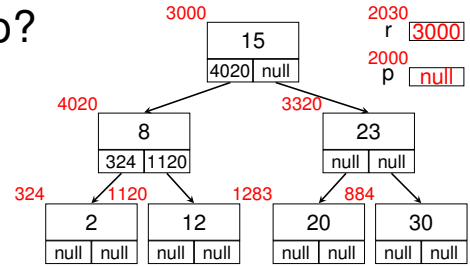


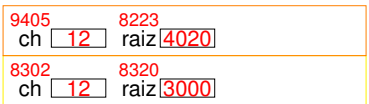


ABP – Busca

E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

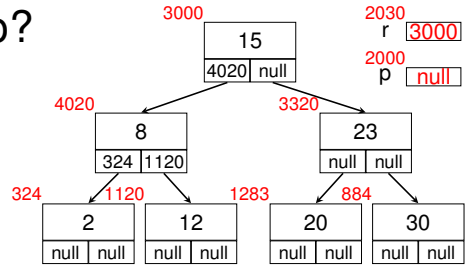


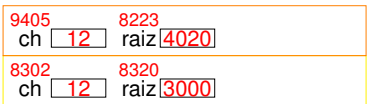


ABP – Busca

E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

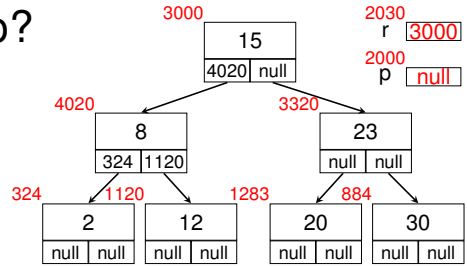


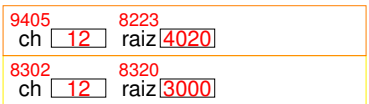


ABP – Busca

E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

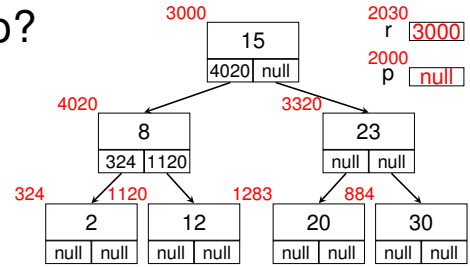




ABP – Busca

E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

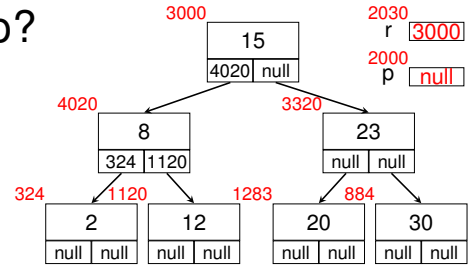


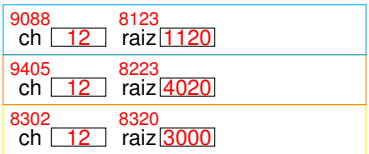
9088		8123	
ch	12	raiz	1120
9405		8223	
ch	12	raiz	4020
8302		8320	
ch	12	raiz	3000

ABP – Busca

E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

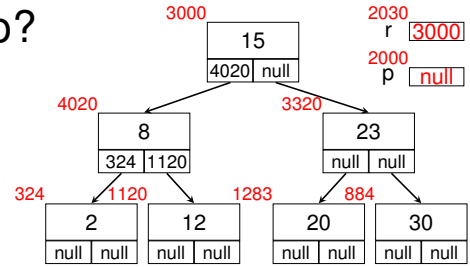


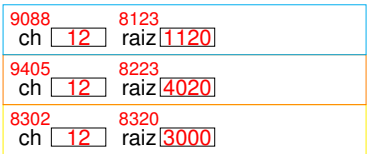


ABP – Busca

E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

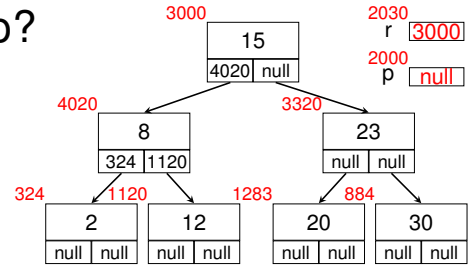


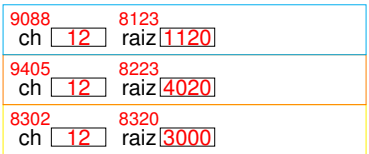


ABP – Busca

E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

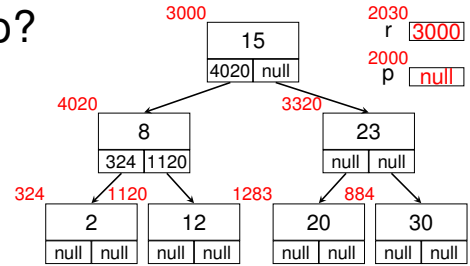


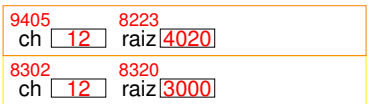


ABP – Busca

E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```

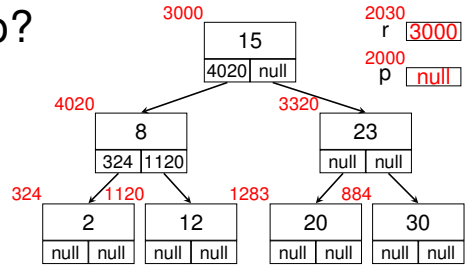




ABP – Busca

E como fica o código para isso?

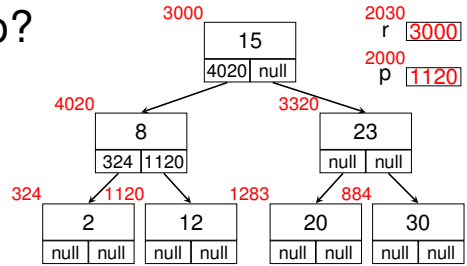
```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```



ABP – Busca

E como fica o código para isso?

```
PONT contem(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return(NULL);
  if (raiz->chave == ch) return(raiz);
  if (raiz->chave>ch)
    return(contem(ch,raiz->esq));
  return(contem(ch,raiz->dir));
}
...
int main(){
  ...
  PONT p = contem(12,r);
}
```



ABP – Contagem dos Elementos

Existem várias maneiras de se percorrer uma árvore binária, em geral envolvendo 3 elementos:

- O nó raiz
- A subárvore à esquerda de cada nó
- A subárvore à direita de cada nó

ABP – Contagem dos Elementos

Algumas possibilidades são

Subárvore esquerda – raiz – subárvore direita

Subárvore direita – raiz – subárvore esquerda

Raiz – subárvore esquerda – subárvore direita

. . .

ABP – Contagem dos Elementos

Em relação a árvores binárias de pesquisa, uma ordem bastante útil é subárvore esquerda – raiz – subárvore direita

- Também chamada de inorder traversal, varredura infixa, ou varredura central
- Nessa varredura, os nós são visitados na ordem crescente das chaves de busca

ABP – Contagem dos Elementos

Vamos então contar o número (n) de nós na árvore:

ABP – Contagem dos Elementos

Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

ABP – Contagem dos Elementos

Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à
esquerda

ABP – Contagem dos Elementos

Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

ABP – Contagem dos Elementos

Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita

ABP – Contagem dos Elementos

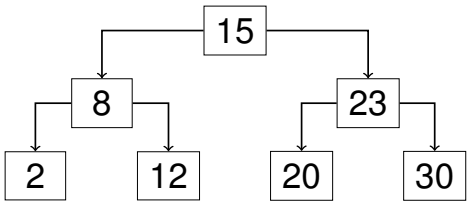
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 0

ABP – Contagem dos Elementos

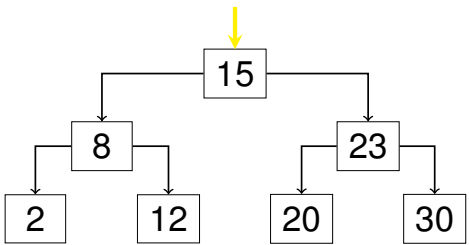
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 0

ABP – Contagem dos Elementos

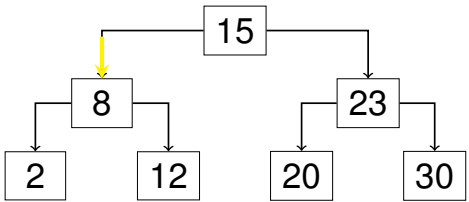
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 0

ABP – Contagem dos Elementos

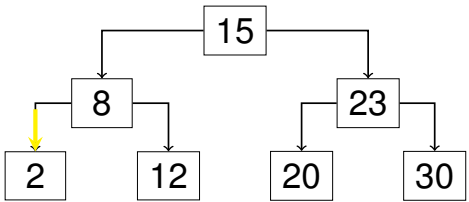
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 0

ABP – Contagem dos Elementos

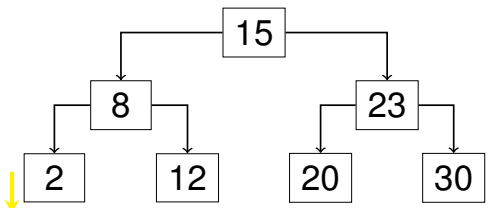
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 0

ABP – Contagem dos Elementos

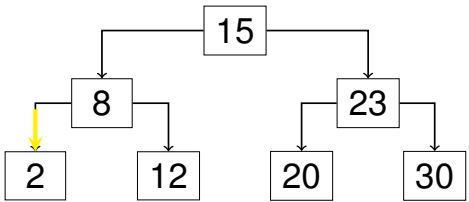
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 1

ABP – Contagem dos Elementos

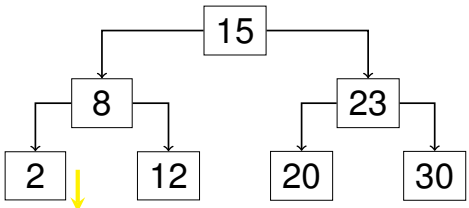
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 1

ABP – Contagem dos Elementos

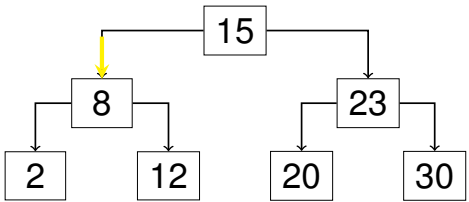
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 2

ABP – Contagem dos Elementos

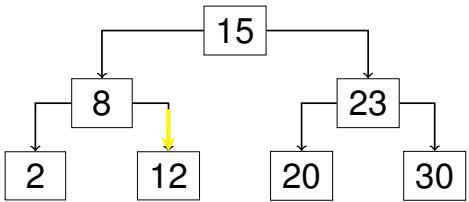
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 2

ABP – Contagem dos Elementos

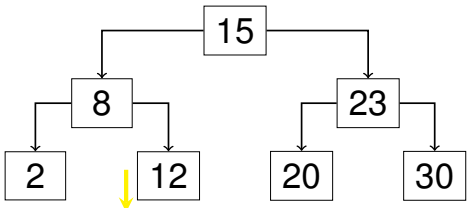
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 2

ABP – Contagem dos Elementos

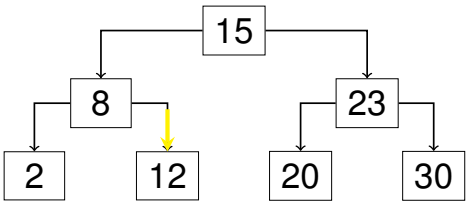
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 3

ABP – Contagem dos Elementos

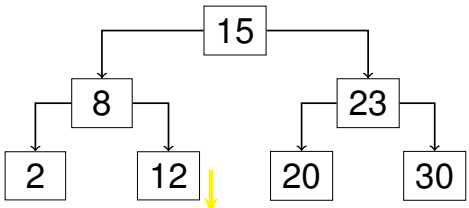
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 3

ABP – Contagem dos Elementos

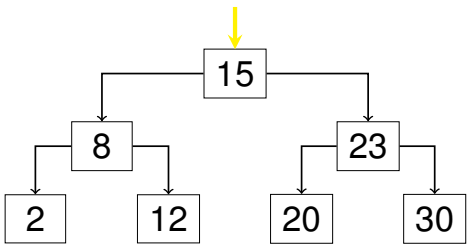
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 4

ABP – Contagem dos Elementos

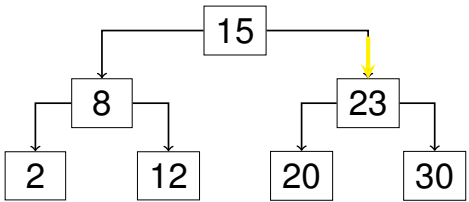
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 4

ABP – Contagem dos Elementos

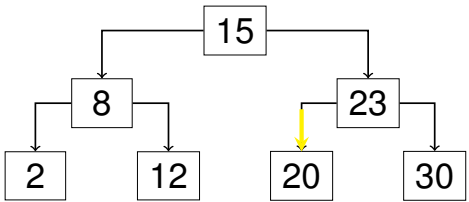
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 4

ABP – Contagem dos Elementos

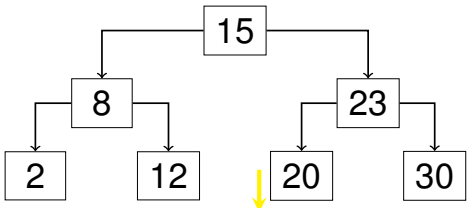
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 4

ABP – Contagem dos Elementos

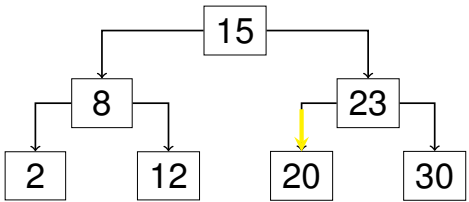
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 5

ABP – Contagem dos Elementos

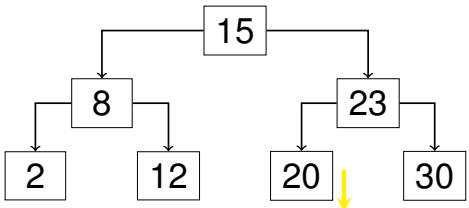
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 5

ABP – Contagem dos Elementos

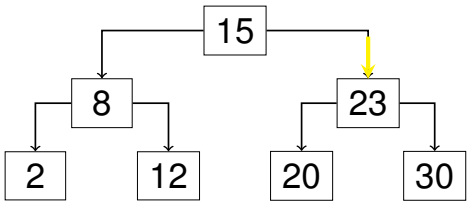
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 6

ABP – Contagem dos Elementos

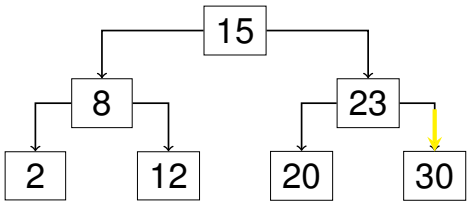
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 6

ABP – Contagem dos Elementos

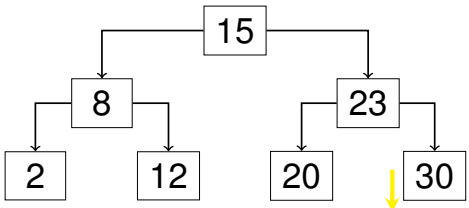
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 6

ABP – Contagem dos Elementos

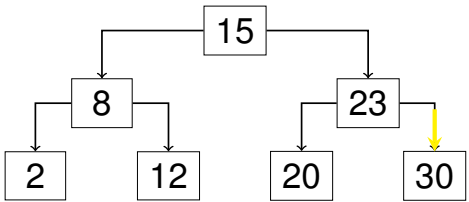
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

Some a subárvore à direita



Contador: 7

ABP – Contagem dos Elementos

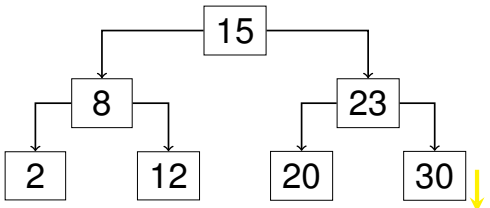
Vamos então contar o número (n) de nós na árvore:

Se não houver raiz, $n = 0$

Conte a subárvore à esquerda

Some a raiz

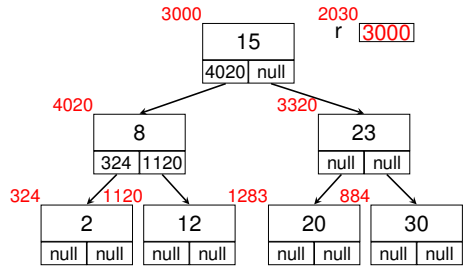
Some a subárvore à direita



Contador: 7

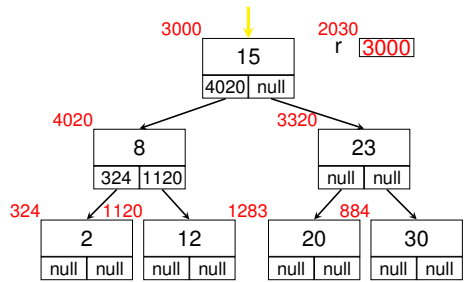
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



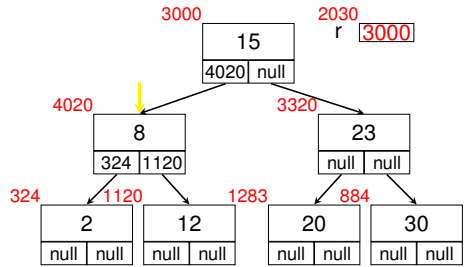
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



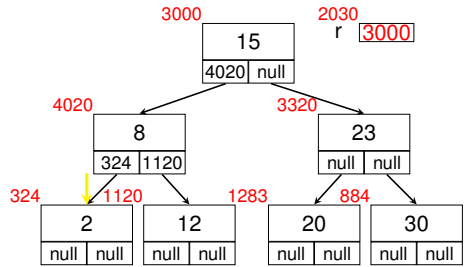
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



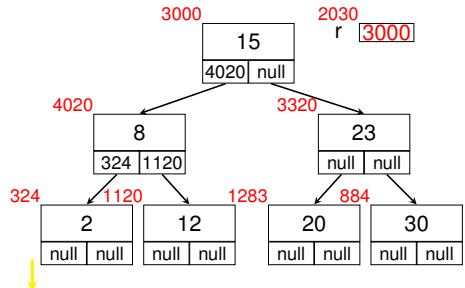
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



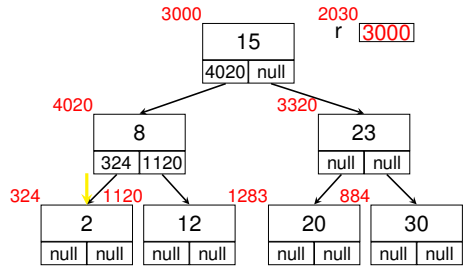
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



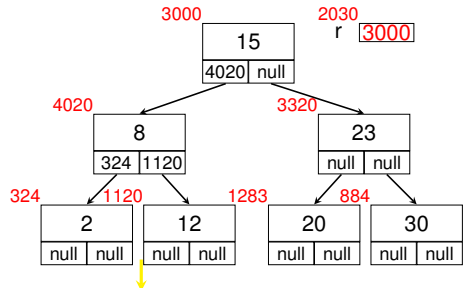
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



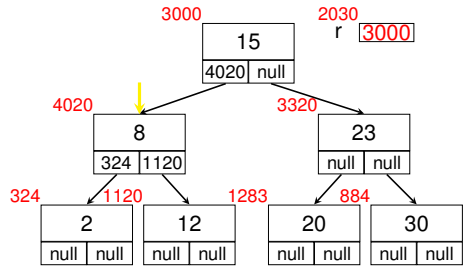
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



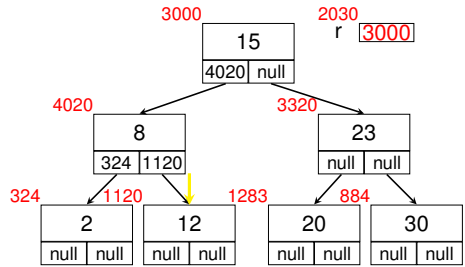
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



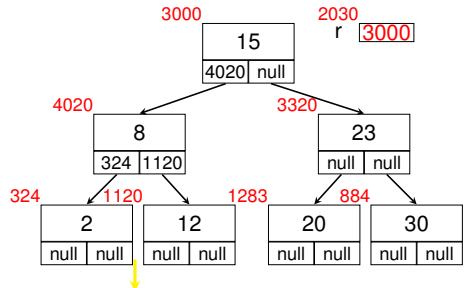
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



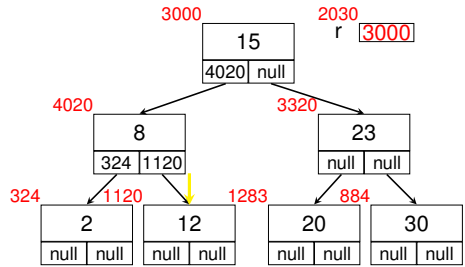
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



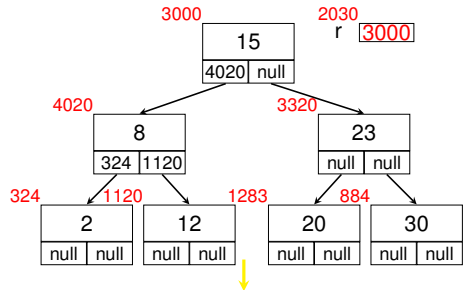
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



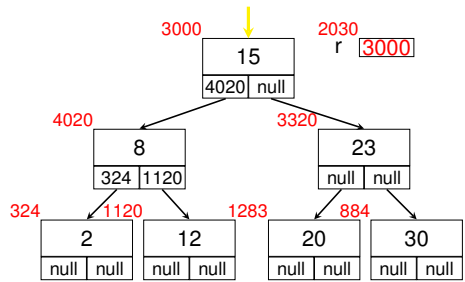
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



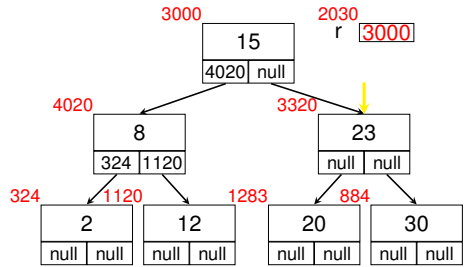
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



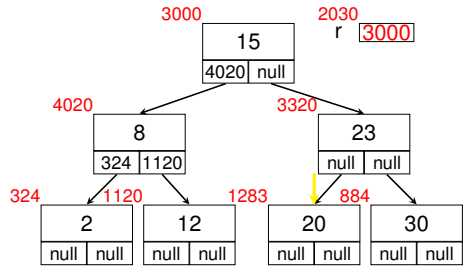
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



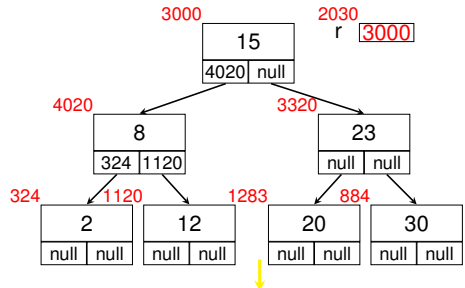
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



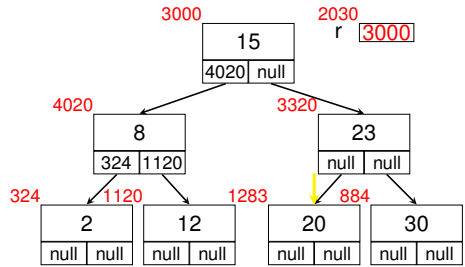
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



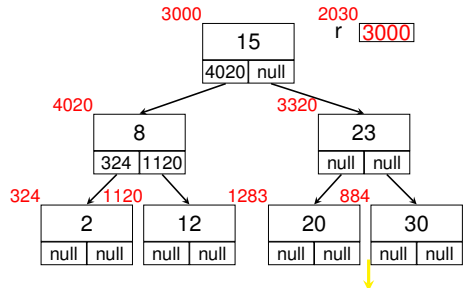
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



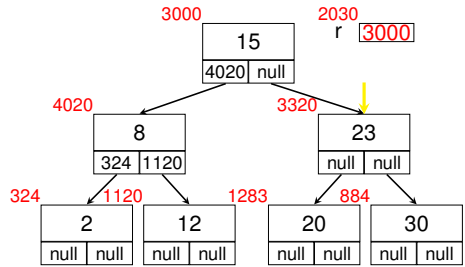
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



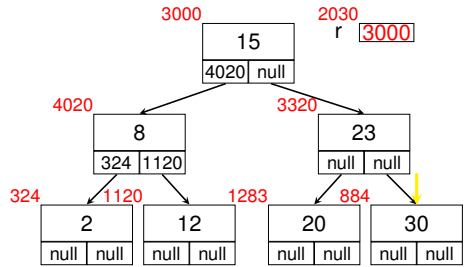
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



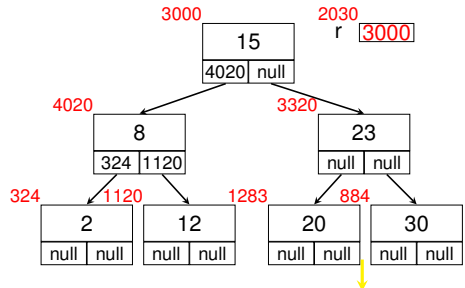
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



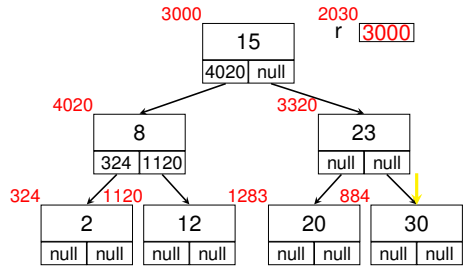
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



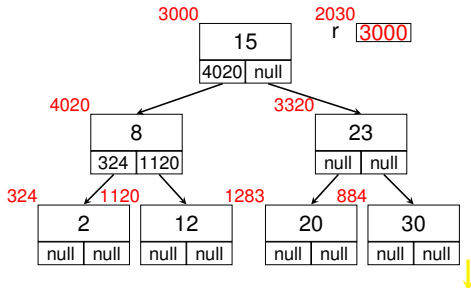
ABP – Contagem dos Elementos

```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



ABP – Contagem dos Elementos

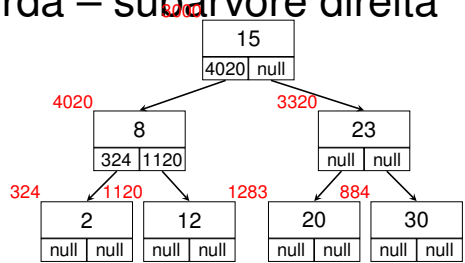
```
int numeroNos(PONT raiz){
  if (!raiz ) return 0;
  return(numeroNos(raiz->esq)
        + 1
        + numeroNos(raiz->dir));
}
...
int main(){
  PONT r = inicializa();
  ...
  printf("%d\n",numeroNos(r));
}
```



ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

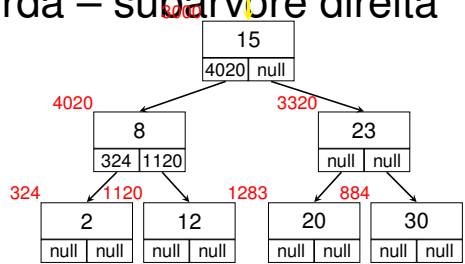


Saída:

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

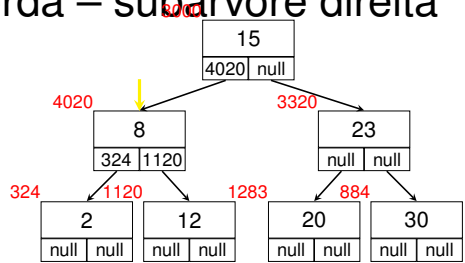


Saída: 15(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

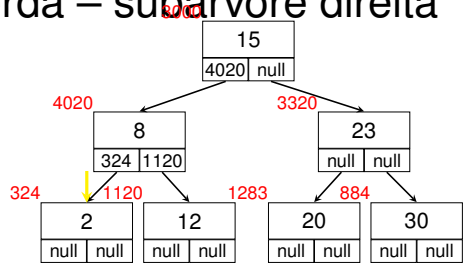


Saída: 15(8(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

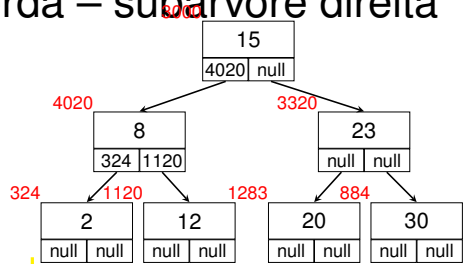


Saída: 15(8(2(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

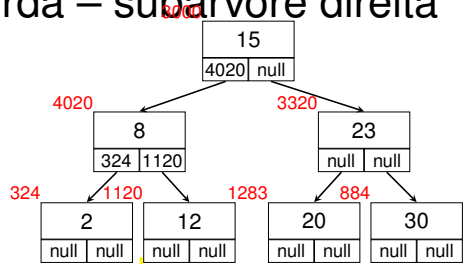
```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```



ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

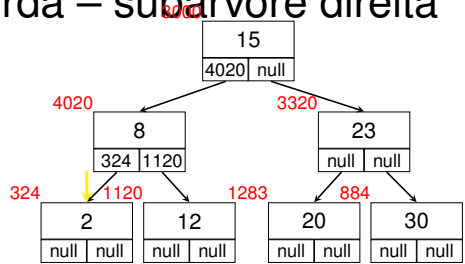
```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```



ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

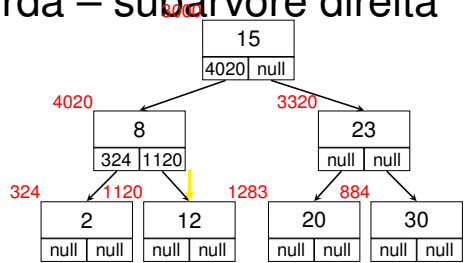


Saída: 15(8(2()

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

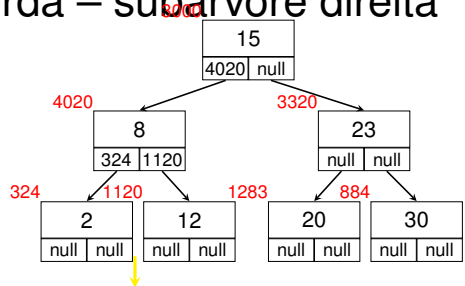


Saída: 15(8(2()12(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

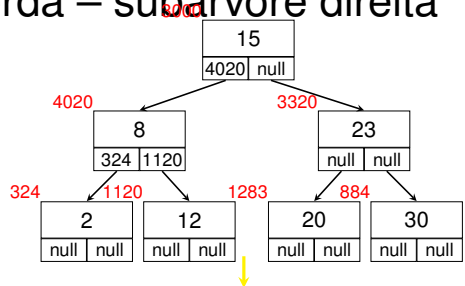
```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```



ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

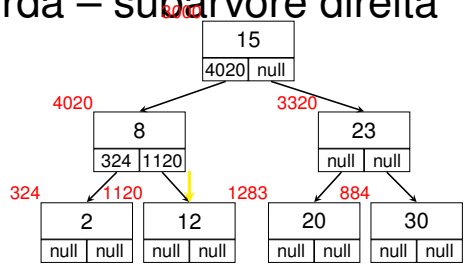


Saída: 15(8(2()12(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

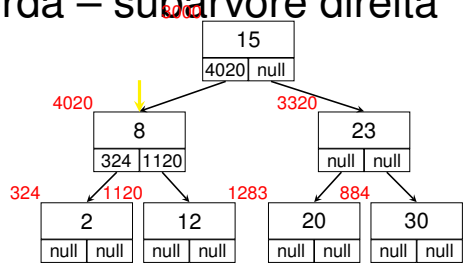


Saída: 15(8(2()12()

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

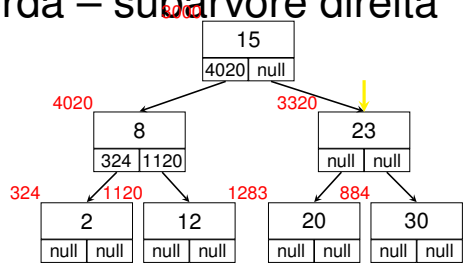


Saída: 15(8(2()12())

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

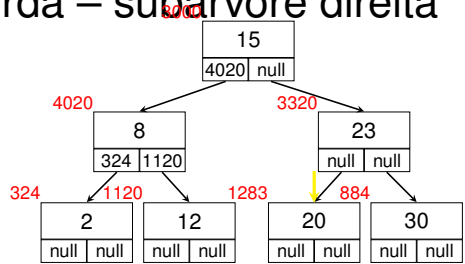


Saída: 15(8(2()12())23(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

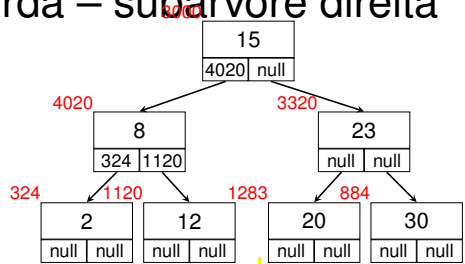


Saída: 15(8(2()12())23(20(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

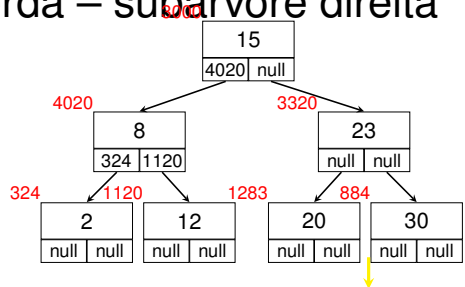
```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```



ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

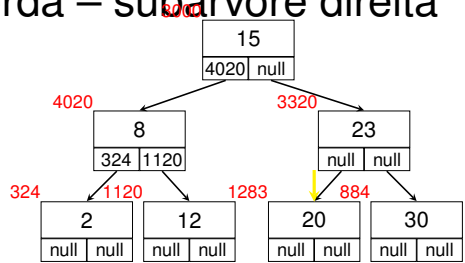


Saída: 15(8(2()12())23(20(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

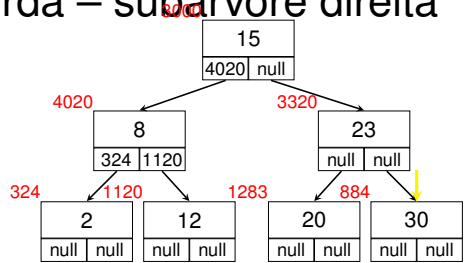


Saída: 15(8(2()12())23(20()

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

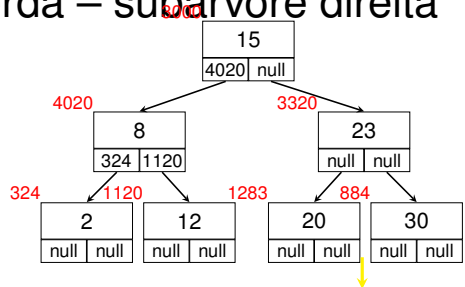


Saída: 15(8(2()12())23(20()30(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

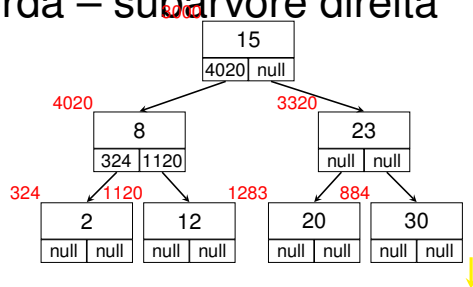


Saída: 15(8(2()12())23(20()30(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

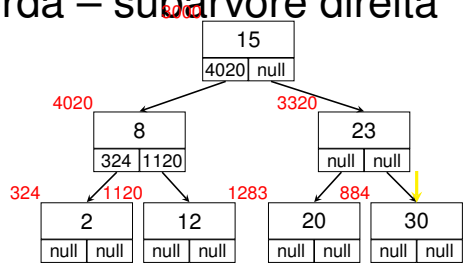


Saída: 15(8(2()12())23(20()30(

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

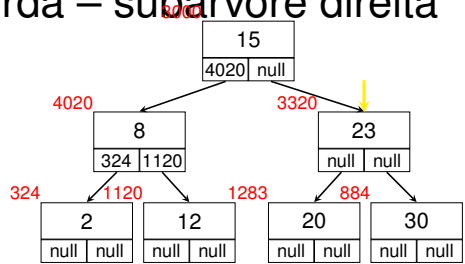


Saída: 15(8(2()12())23(20()30()

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```

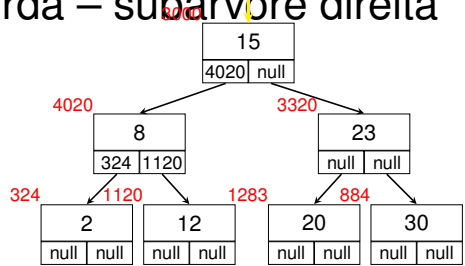


Saída: 15(8(2()12())23(20()30())

ABP – Leitura

Para imprimir a árvore, pode ser conveniente usarmos a ordem raiz – subárvore esquerda – subárvore direita

```
void exibirArvore(PONT raiz){
  if (raiz != NULL) {
    printf("%i",raiz->chave);
    printf("(");
    exibirArvore(raiz->esq);
    exibirArvore(raiz->dir);
    printf(")");
  }
}
```



Saída: 15(8(2()12())23(20()30()))

Árvore Binária de Pesquisa

Note que, para qualquer procedimento, podemos usar qualquer nó da árvore como raiz

Árvore Binária de Pesquisa

Note que, para qualquer procedimento, podemos usar qualquer nó da árvore como raiz

- Podemos então contar nós ou imprimir qualquer subárvore

Árvore Binária de Pesquisa

Note que, para qualquer procedimento, podemos usar qualquer nó da árvore como raiz

- Podemos então contar nós ou imprimir qualquer subárvore
- Só não podemos incluir e excluir nós sem ter começado da raiz. Do contrário poderemos perder a ordem dos nós

AULA 17
ESTRUTURA DE DADOS

Árvores Binárias de Pesquisa

Norton T. Roman & Luciano A. Digiampietri