AULA e07b
Algoritmos e Estruturas de Dados I

Árvores Binárias - Percursos (parte II)

Luciano Antonio Digiampietri

Pré-ordem: implementação iterativa

As funções recursivas utilizavam a **pilha de execução** para gerenciar os percursos realizados.

Pré-ordem: implementação iterativa

As funções recursivas utilizavam a pilha de execução para gerenciar os percursos realizados.

Para realizarmos o percurso em pré-ordem de maneira não-recursiva precisaremos utilizar uma estrutura auxiliar que funcione como a pilha de execução (isto é, utilizaremos uma pilha).

Pré-ordem: implementação iterativa

Ideia geral:

Pré-ordem: implementação iterativa

Ideia geral:

Criamos uma pilha e inserimos o nó raiz;

Pré-ordem: implementação iterativa

Ideia geral:

Criamos uma pilha e inserimos o nó raiz;

Enquanto a pilha não estiver vazia:

Pré-ordem: implementação iterativa

Ideia geral:

Criamos uma pilha e inserimos o nó raiz;

Enquanto a pilha não estiver vazia:

- retiramos o nó de seu topo;

Pré-ordem: implementação iterativa

Ideia geral:

Criamos uma pilha e inserimos o nó raiz;

Enquanto a pilha não estiver vazia:

- **retiramos o nó de seu topo;**
- **imprimimos seu valor;**

Pré-ordem: implementação iterativa

Ideia geral:

Criamos uma pilha e inserimos o nó raiz;

Enquanto a pilha não estiver vazia:

- **retiramos o nó de seu topo;**
- **imprimimos seu valor;**
- **inserimos seus filhos (se existirem) na pilha.**

Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){









}
```

Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;









}
```

Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);









}
```

Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);









}
```

Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {




  }
  printf("\n");
}
```

Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);



  }
  printf("\n");
}
```

Pré-ordem: implementação iterativa

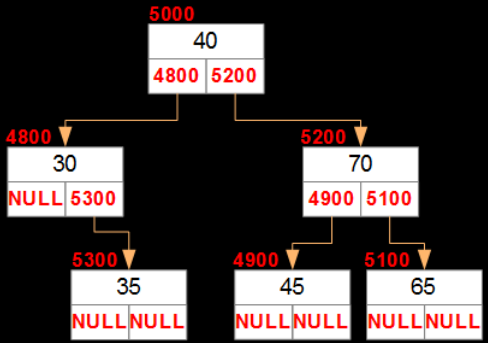
```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);


  }
  printf("\n");
}
```

Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

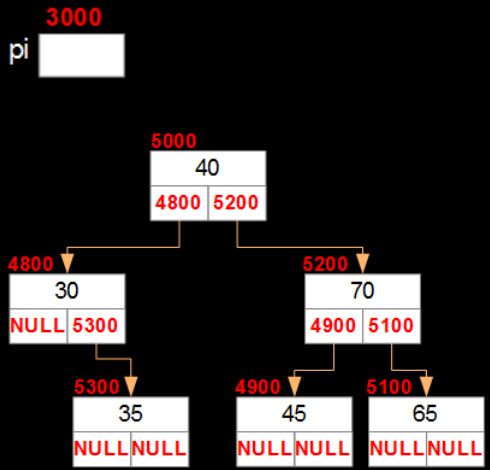
Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```



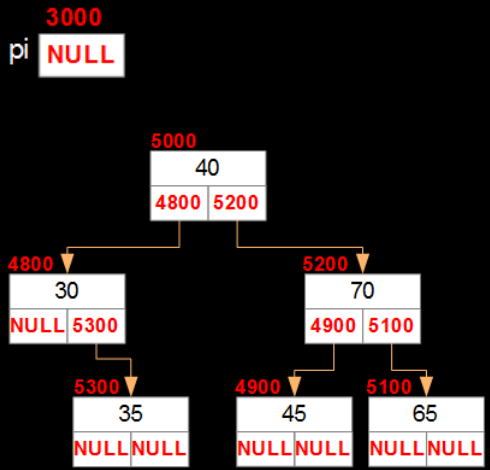
Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```



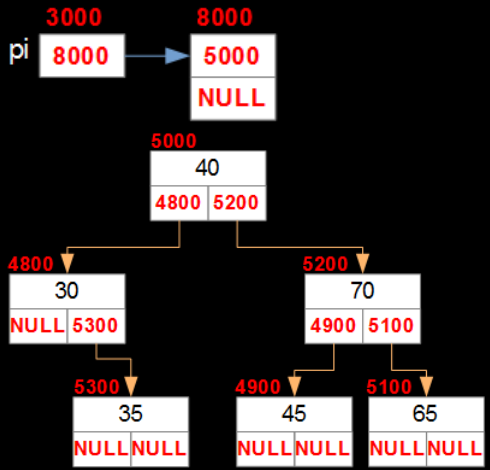
Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

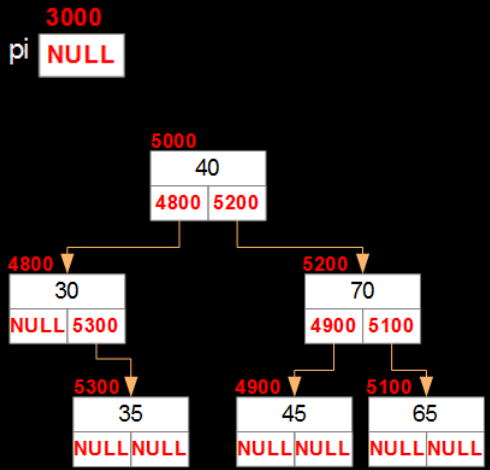


Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

atual: 5000

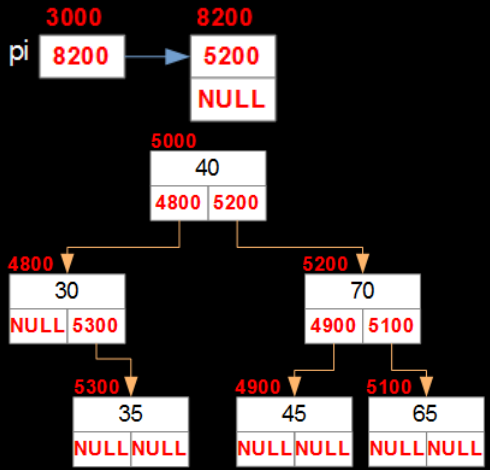
$ 40



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

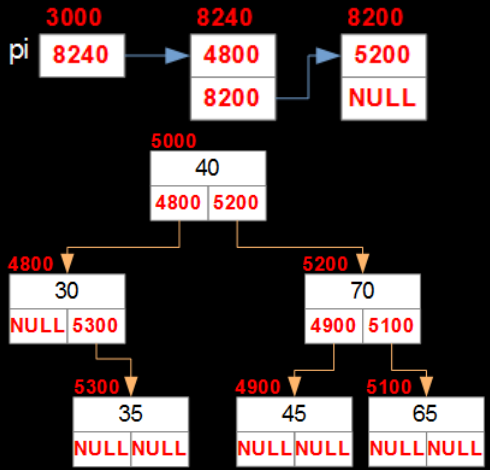
```
atual: 5000
$ 40
```



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

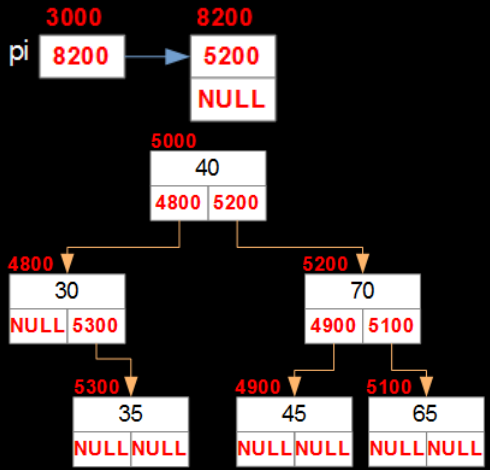
```
atual: 5000
$ 40
```



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

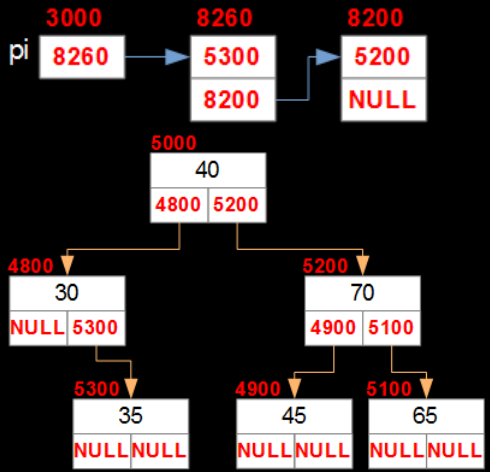
```
atual: 4800
$ 40 30
```



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

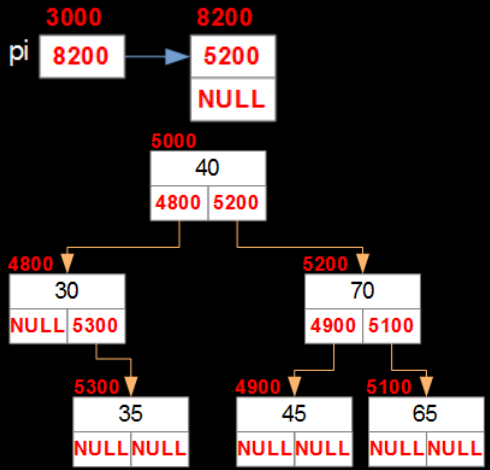
```
atual: 4800
$ 40 30
```



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

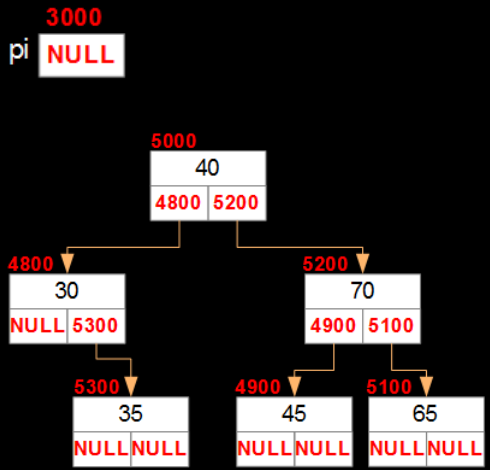
```
atual: 5300
$ 40 30 35
```



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

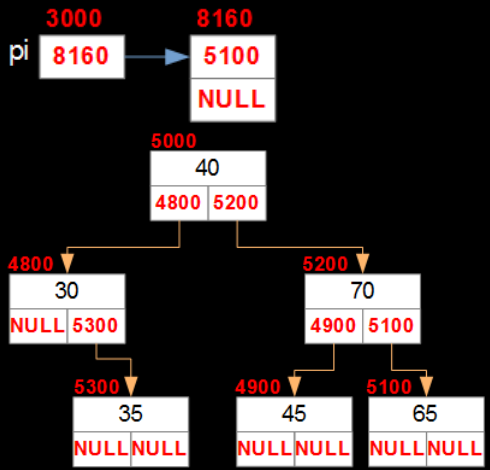
```
atual: 5200
$ 40 30 35 70
```



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

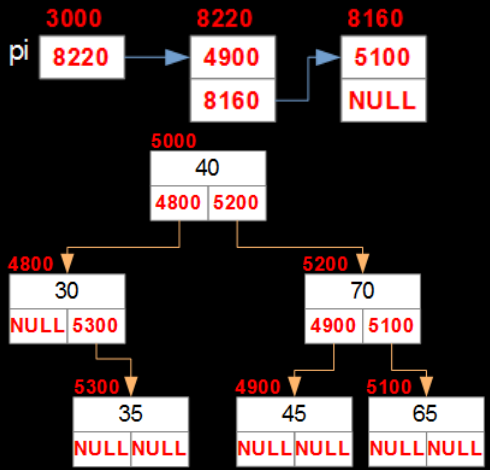
```
atual: 5200
$ 40 30 35 70
```



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

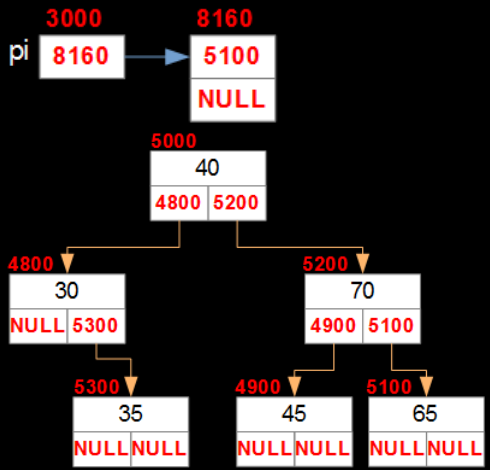
```
atual: 5200
$ 40 30 35 70
```



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

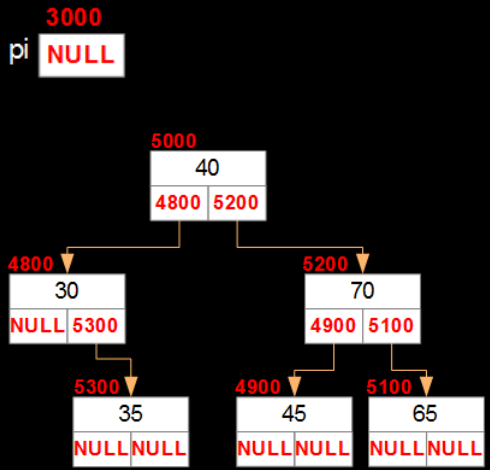
```
atual: 4900
$ 40 30 35 70 45
```



Pré-ordem: implementação iterativa

```
void preOrdemNaoRecursivo(PONT raiz){
  if (!raiz) return;
  PILHA pi;
  inicializarPilha(&pi);
  push(raiz, &pi);
  PONT atual;
  while (!isEmpty(&pi)) {
     atual = pop(&pi);
     printf("%i ",atual->chave);
     if (atual->dir) push(atual->dir, &pi);
     if (atual->esq) push(atual->esq, &pi);
  }
  printf("\n");
}
```

```
   atual: 5100
   $ 40 30 35 70 45 65
```



Em nível: implementação iterativa

Precisamos imprimir os elementos um nível de cada vez.

Em nível: implementação iterativa

Precisamos imprimir os elementos um nível de cada vez.

De cima para baixo e da esquerda para a direita;

Em nível: implementação iterativa

Precisamos imprimir os elementos um nível de cada vez.

De cima para baixo e da esquerda para a direita;

Só podemos imprimir elementos do próximo nível após imprimir todos os elementos do nível atual;

Em nível: implementação iterativa

Precisamos imprimir os elementos um nível de cada vez.

De cima para baixo e da esquerda para a direita;

Só podemos imprimir elementos do próximo nível após imprimir todos os elementos do nível atual;

Assim, ao visitar um elemento temos que enfileirar seus filhos para serem exibidos após os demais elementos que já estiverem na fila.

Em nível: implementação iterativa

Ideia geral:

Em nível: implementação iterativa

Ideia geral:

Criamos uma fila e inserimos o nó raiz;

Em nível: implementação iterativa

Ideia geral:

Criamos uma fila e inserimos o nó raiz;

Enquanto a fila não estiver vazia:

Em nível: implementação iterativa

Ideia geral:

Criamos uma fila e inserimos o nó raiz;

Enquanto a fila não estiver vazia:

- retiramos o nó de seu início;

Em nível: implementação iterativa

Ideia geral:

Criamos uma fila e inserimos o nó raiz;

Enquanto a fila não estiver vazia:

- **retiramos o nó de seu início;**
- **imprimimos seu valor;**

Em nível: implementação iterativa

Ideia geral:

Criamos uma fila e inserimos o nó raiz;

Enquanto a fila não estiver vazia:

- **retiramos o nó de seu início;**
- **imprimimos seu valor;**
- **inserimos seus filhos (se existirem) na fila.**

Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {









}
```

Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;









}
```

Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);









}
```

Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);









}
```

Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){




  }
  printf("\n");
}
```

Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);



  }
  printf("\n");
}
```

Percurso em nível

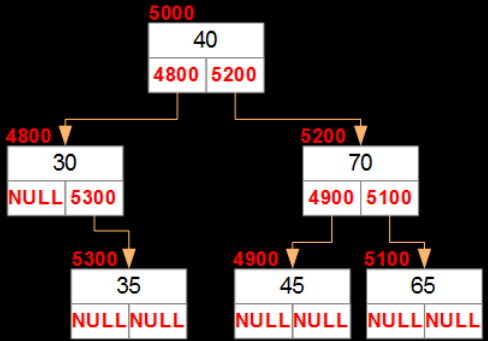
```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);


  }
  printf("\n");
}
```

Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq) entrarFila(atual->esq, &f);
    if(atual->dir) entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

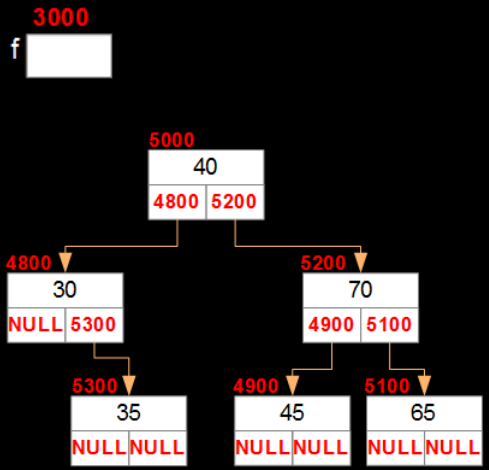
Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```



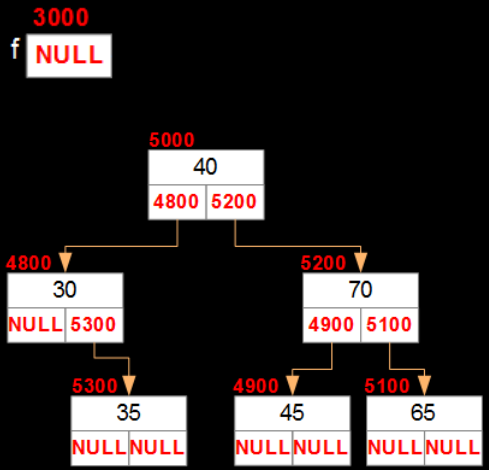
Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```



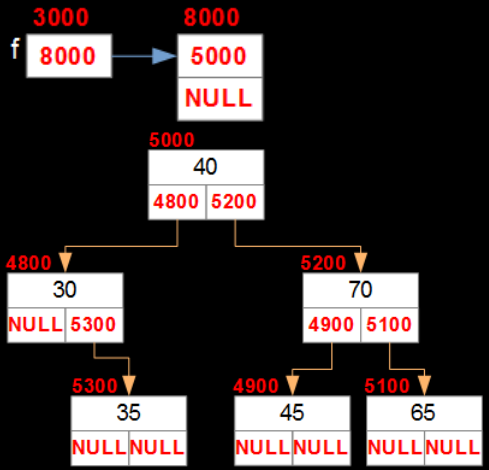
Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```



Percurso em nível

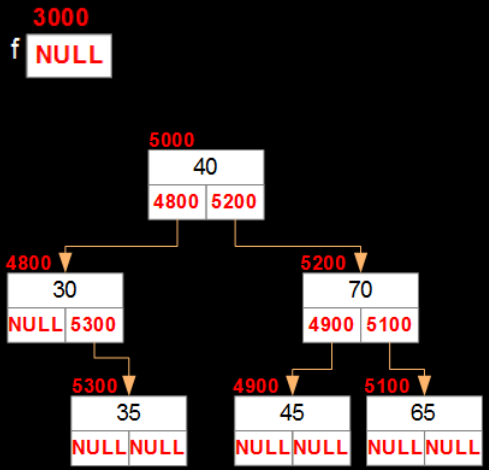
```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

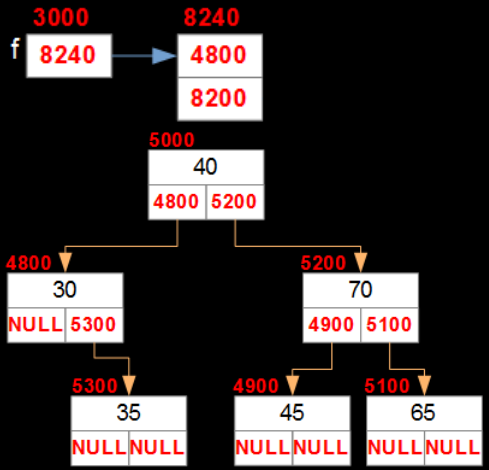
```
atual: 5000
$ 40
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

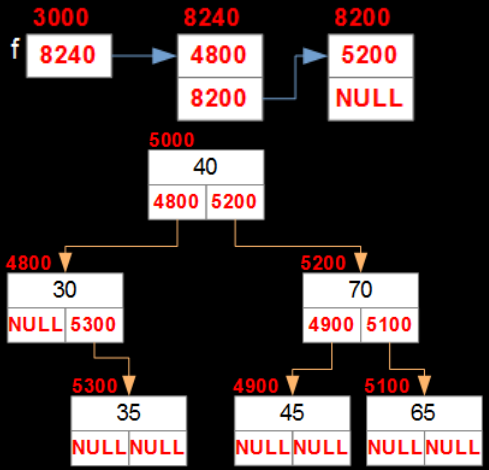
```
atual: 5000
$ 40
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

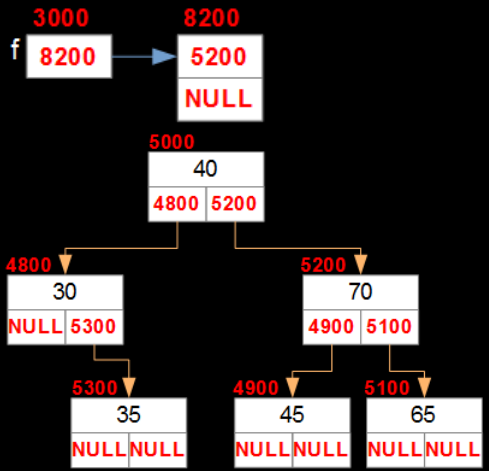
```
atual: 5000
$ 40
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

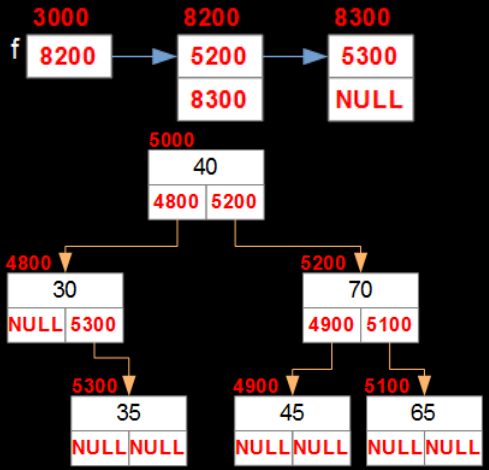
```
atual: 4800
$ 40 30
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

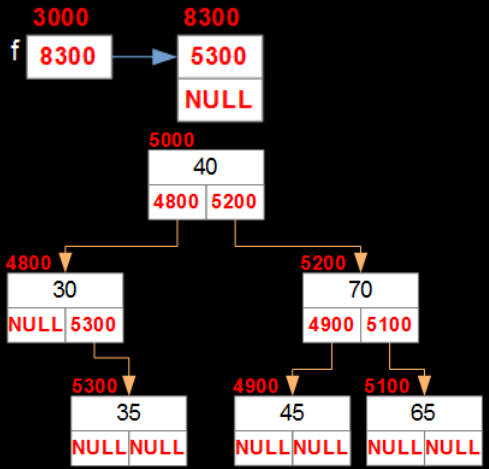
```
atual: 4800
$ 40 30
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

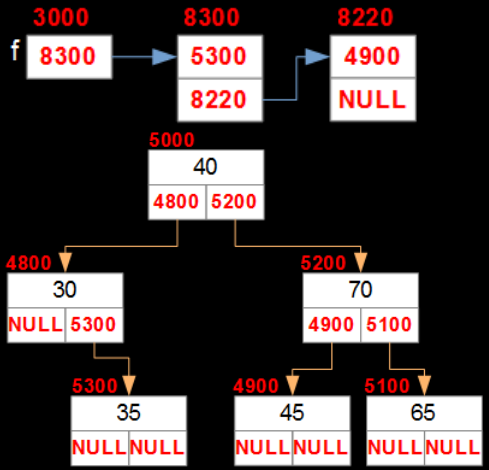
```
atual: 5200
$ 40 30 70
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

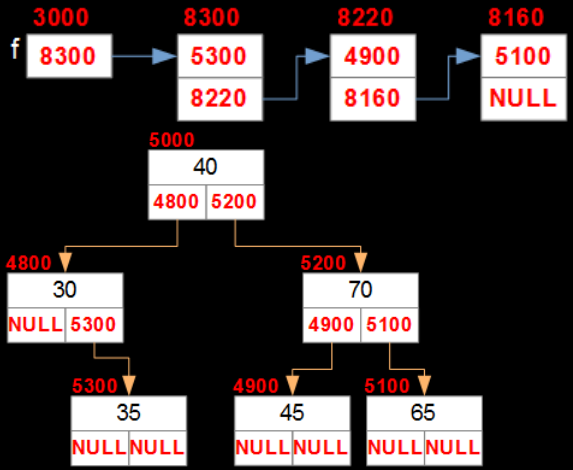
```
atual: 5200
$ 40 30 70
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
        entrarFila(atual->esq, &f);
    if(atual->dir)
        entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

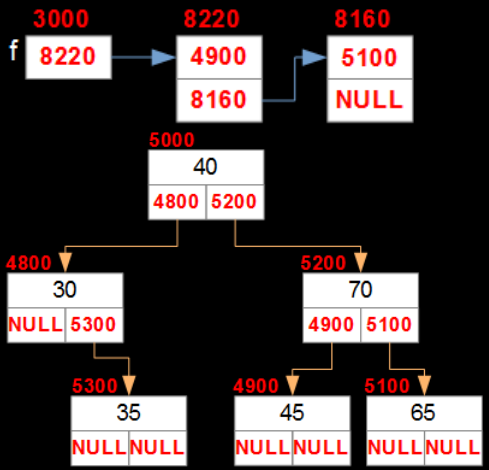
```
atual: 5200
$ 40 30 70
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

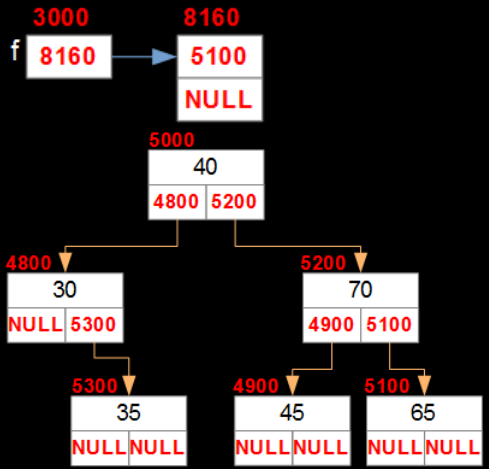
```
atual: 5300
$ 40 30 70 35
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

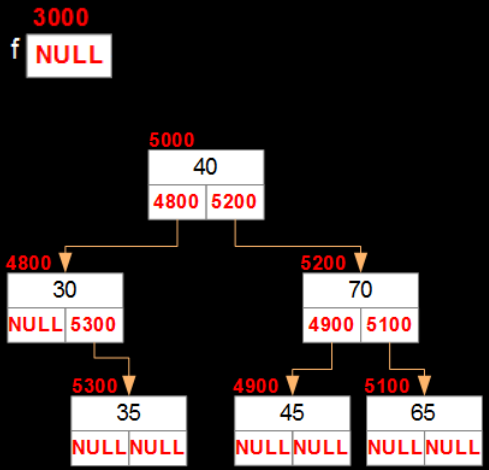
```
atual: 4900
$ 40 30 70 35 45
```



Percurso em nível

```
void exibirArvoreEmNivel(PONT raiz) {
  if (!raiz) return;
  FILA f;
  inicializarFila(&f);
  entrarFila(raiz,&f);
  PONT atual;
  while((f.inicio)){
    atual = sairFila(&f);
    printf("%i ", atual->chave);
    if(atual->esq)
         entrarFila(atual->esq, &f);
    if(atual->dir)
         entrarFila(atual->dir, &f);
  }
  printf("\n");
}
```

```
atual: 5100
$ 40 30 70 35 45 65
```



AULA e07b
Algoritmos e Estruturas de Dados I

Árvores Binárias - Percursos (parte II)

Luciano Antonio Digiampietri