AULA 19
ESTRUTURA DE DADOS

Árvores N-árias

Norton T. Roman & Luciano A. Digiampietri

Árvores N-árias

Algumas vezes, a árvore binária não é a melhor escolha

- Se lidar com grandes volumes de dados, sua profundidade cresce
- Dependendo da modelagem do problema, cada nó precisa ter um número variado de filhos, de 0 a n, com $n > 2$

Árvores N-árias

Que fazer?

Árvores N-árias

Que fazer?

Se precisamos de mais filhos, mais filhos é o que teremos – árvore n-ária

Árvores N-árias

Que fazer?

Se precisamos de mais filhos, mais filhos é o que teremos – árvore n-ária

Definição 1: Uma árvore n-ária é uma árvore em que cada nó pode ter até *n* filhos

Árvores N-árias

Que fazer?

Se precisamos de mais filhos, mais filhos é o que teremos – árvore n-ária

Definição 1: Uma árvore n-ária é uma árvore em que cada nó pode ter até *n* filhos

Definição 2: Uma árvore n-ária é uma árvore em que cada nó pode ter um número *arbitrário* de filhos

Árvores N-árias

Trata-se então de uma generalização das árvores binárias

Árvores N-árias

Trata-se então de uma generalização das árvores binárias

E qual definição usar?

Árvores N-árias

Trata-se então de uma generalização das árvores binárias

E qual definição usar?

- A que melhor se adaptar ao problema modelado

Árvores N-árias

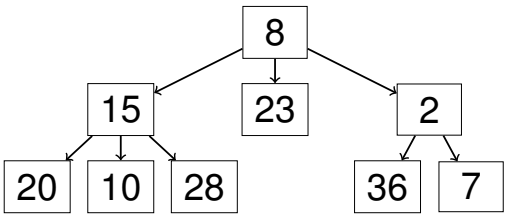
Trata-se então de uma generalização das árvores binárias

E qual definição usar?

- A que melhor se adaptar ao problema modelado
- Nessa aula, seguiremos o modelo mais geral, seguindo a Definição 2

Árvores N-árias

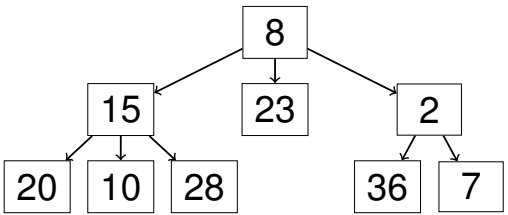
Suponha que queremos modelar a seguinte árvore:



Árvores N-árias

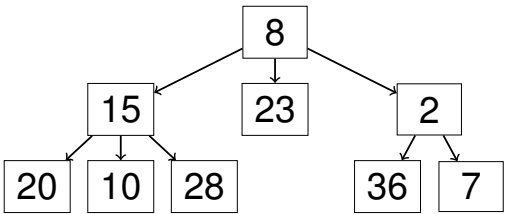
Suponha que queremos modelar a seguinte árvore:

Como seria a representação do nó?



Árvores N-árias

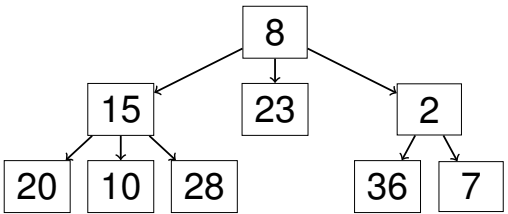
Antes, repare que não há ordem nos nós...



Árvores N-árias

Antes, repare que não há ordem nos nós...

Mas e como fica sua representação?

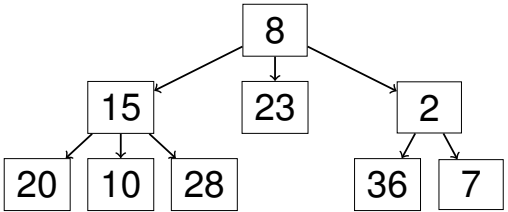


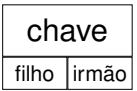
Árvores N-árias

Antes, repare que não há ordem nos nós...

Mas e como fica sua representação?

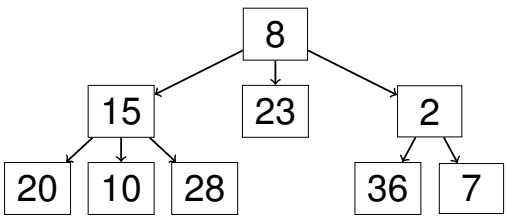
Uma possibilidade é:





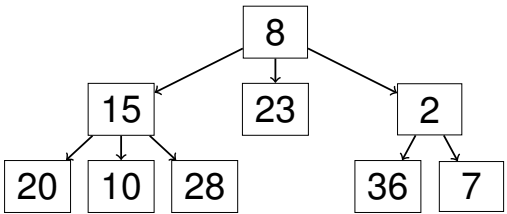
Árvores N-árias

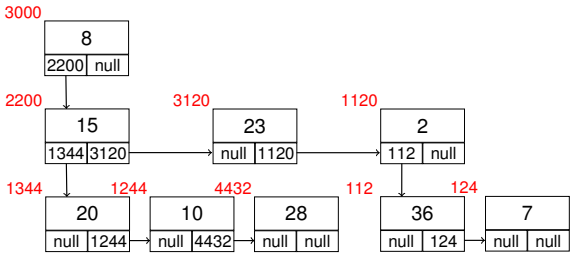
Então, nossa árvore fica...



Árvores N-árias

Então, nossa árvore fica...

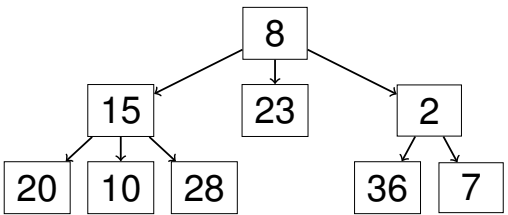




Árvores N-árias

Veremos código para:

- Inicialização
- Inserção
- Exibição
- Busca



Árvores N-árias

Antes de mais nada, vamos definir as estruturas necessárias...

```
#include <stdio.h>
#include <stdlib.h>
#define true 1
#define false 0

typedef int bool;
typedef int TIPOCHAVE;
```

```
typedef struct no {
  TIPOCHAVE chave;
  /*Aqui vão outros dados*/
  struct no *primFilho;
  struct no *proxIrmao;
} NO;

typedef NO* PONT;
```

Árvores N-árias – Inicialização

Para inicializar, temos que primeiro criar um nó na memória

```
PONT criaNovoNo(TIPOCHAVE ch){
  PONT novo =
     (PONT)malloc(sizeof(NO));
  novo->primFilho = NULL;
  novo->proxIrmao = NULL;
  novo->chave = ch;
  return(novo);
}
```

```
PONT inicializa(TIPOCHAVE ch) {
  return(criaNovoNo(ch));
}

...

int main() {
  PONT r = inicializa(8);
}
```

Árvores N-árias – Inserção

```
bool insere(PONT raiz, TIPOCHAVE novaChave, TIPOCHAVE chavePai){
  PONT pai = buscaChave(chavePai,raiz);
  if (!pai) return(false);
  PONT filho = criaNovoNo(novaChave);
  PONT p = pai->primFilho;
  if (!p) pai->primFilho = filho;
  else {
    while (p->proxIrmao)
      p = p->proxIrmao;
    p->proxIrmao = filho;
  }
  return(true);
}
```

Árvores N-árias – Inserção

```
bool insere(PONT raiz, TIPOCHAVE novaChave, TIPOCHAVE chavePai){
  PONT pai = buscaChave(chavePai,raiz);
  if (!pai) return(false);
  PONT filho = criaNovoNo(novaChave);
  PONT p = pai->primFilho;
  if (!p) pai->primFilho = filho;
  else {
    while (p->proxIrmao)
      p = p->proxIrmao;
    p->proxIrmao = filho;
  }
  return(true);
}
```

Verificamos se o pai existe

Árvores N-árias – Inserção

```
bool insere(PONT raiz, TIPOCHAVE novaChave, TIPOCHAVE chavePai){
  PONT pai = buscaChave(chavePai,raiz);
  if (!pai) return(false);
  PONT filho = criaNovoNo(novaChave);
  PONT p = pai->primFilho;
  if (!p) pai->primFilho = filho;
  else {
    while (p->proxIrmao)
      p = p->proxIrmao;
    p->proxIrmao = filho;
  }
  return(true);
}
```

Criamos o nó para o filho

Árvores N-árias – Inserção

```
bool insere(PONT raiz, TIPOCHAVE novaChave, TIPOCHAVE chavePai){
  PONT pai = buscaChave(chavePai,raiz);
  if (!pai) return(false);
  PONT filho = criaNovoNo(novaChave);
  PONT p = pai->primFilho;
  if (!p) pai->primFilho = filho;
  else {
    while (p->proxIrmao)
      p = p->proxIrmao;
    p->proxIrmao = filho;
  }
  return(true);
}
```

Verificamos o primogênito desse pai

Árvores N-árias – Inserção

```
bool insere(PONT raiz, TIPOCHAVE novaChave, TIPOCHAVE chavePai){
  PONT pai = buscaChave(chavePai,raiz);
  if (!pai) return(false);
  PONT filho = criaNovoNo(novaChave);
  PONT p = pai->primFilho;
  if (!p) pai->primFilho = filho;
  else {
    while (p->proxIrmao)
      p = p->proxIrmao;
    p->proxIrmao = filho;
  }
  return(true);
}
```

Se não houver primogênito, o novo nó é o primeiro filho

Árvores N-árias – Inserção

```
bool insere(PONT raiz, TIPOCHAVE novaChave, TIPOCHAVE chavePai){
  PONT pai = buscaChave(chavePai,raiz);
  if (!pai) return(false);
  PONT filho = criaNovoNo(novaChave);
  PONT p = pai->primFilho;
  if (!p) pai->primFilho = filho;
  else {
    while (p->proxIrmao)
      p = p->proxIrmao;
    p->proxIrmao = filho;
  }
  return(true);
}
```

Do contrário, vamos ao último filho

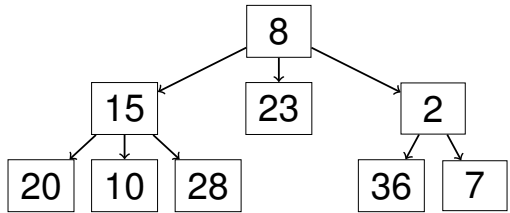
Árvores N-árias – Inserção

```
bool insere(PONT raiz, TIPOCHAVE novaChave, TIPOCHAVE chavePai){
  PONT pai = buscaChave(chavePai,raiz);
  if (!pai) return(false);
  PONT filho = criaNovoNo(novaChave);
  PONT p = pai->primFilho;
  if (!p) pai->primFilho = filho;
  else {
    while (p->proxIrmao)
      p = p->proxIrmao;
    p->proxIrmao = filho;
  }
  return(true);
}
```

Colocamos o novo nó como caçula

Árvores N-árias – Exibição

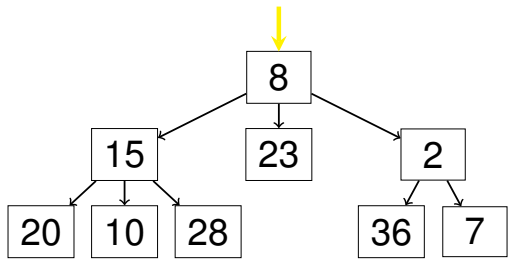
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída:

Árvores N-árias – Exibição

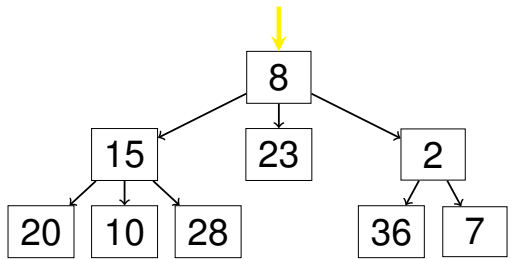
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída:

Árvores N-árias – Exibição

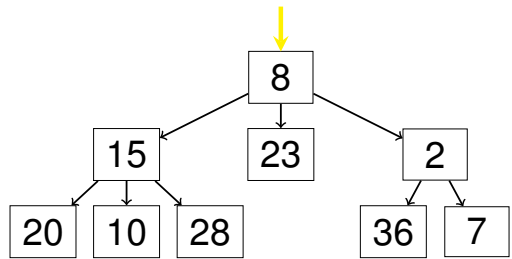
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída:

Árvores N-árias – Exibição

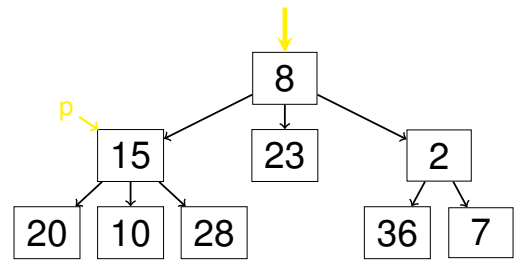
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(

Árvores N-árias – Exibição

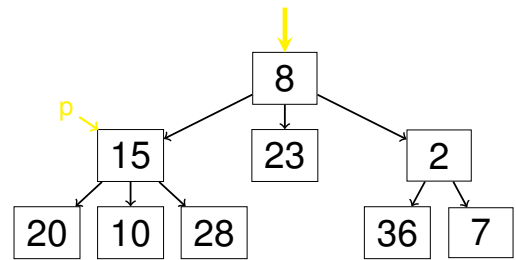
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(

Árvores N-árias – Exibição

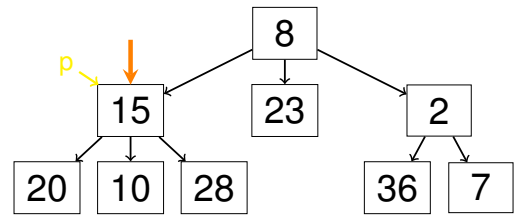
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(

Árvores N-árias – Exibição

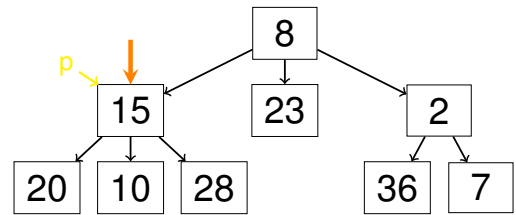
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(

Árvores N-árias – Exibição

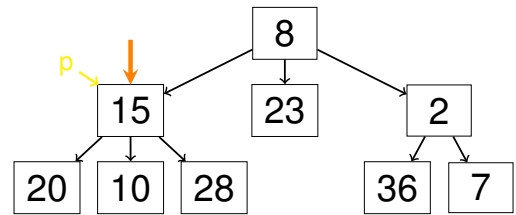
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(

Árvores N-árias – Exibição

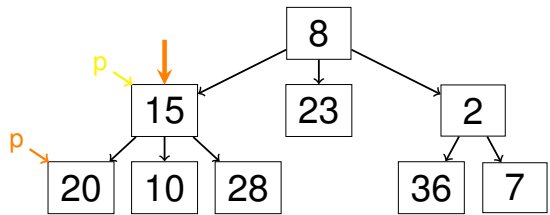
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(

Árvores N-árias – Exibição

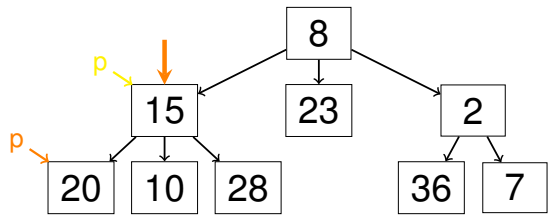
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(

Árvores N-árias – Exibição

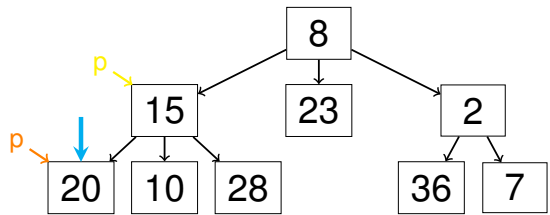
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(

Árvores N-árias – Exibição

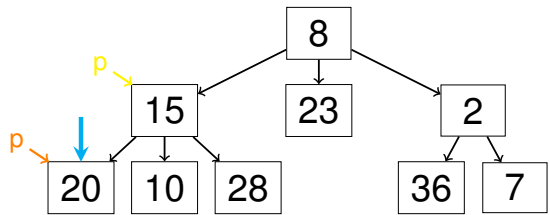
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(

Árvores N-árias – Exibição

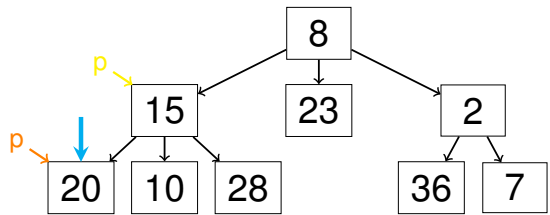
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(

Árvores N-árias – Exibição

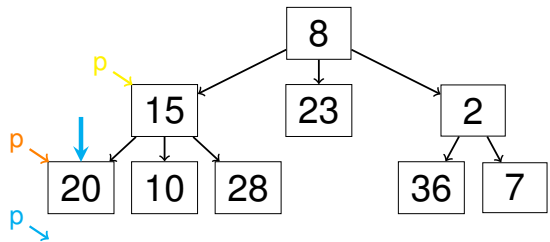
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(20(

Árvores N-árias – Exibição

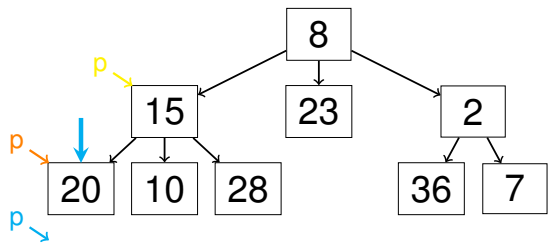
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(20(

Árvores N-árias – Exibição

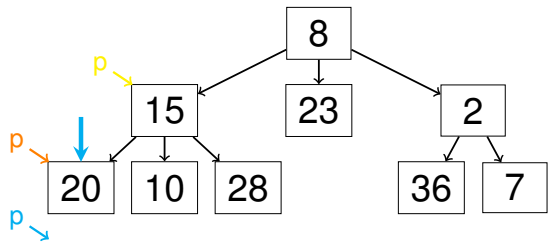
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(20(

Árvores N-árias – Exibição

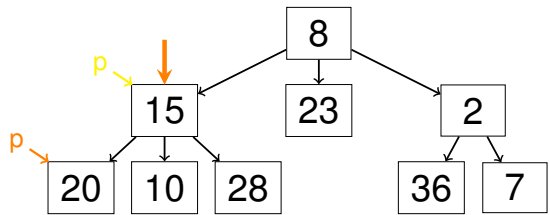
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(20()

Árvores N-árias – Exibição

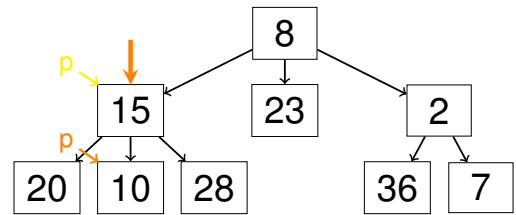
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(20()

Árvores N-árias – Exibição

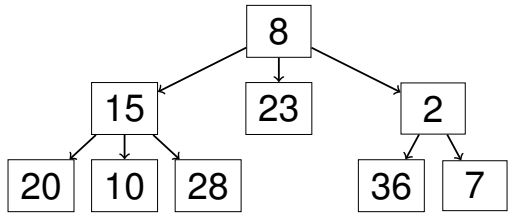
```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Saída: 8(15(20()

Árvores N-árias – Exibição

```
void exibirArvore(PONT raiz){
  if (raiz == NULL) return;
  printf("%d(",raiz->chave);
  PONT p = raiz->primFilho;
  while (p) {
    exibirArvore(p);
    p = p->proxIrmao;
  }
  printf(")");
}
```



Após o fim do algoritmo, teremos

Saída: 8(15(20()10()28())23()2(36()7()))

Árvores N-árias – Busca

```
PONT buscaChave(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return NULL;
  if (raiz->chave == ch) return raiz;
  PONT p = raiz->primFilho;
  while(p) {
    PONT resp = buscaChave(ch, p);
    if (resp) return(resp);
    p = p->proxIrmao;
  }
  return(NULL);
}
```

Árvores N-árias – Busca

```
PONT buscaChave(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return NULL;
  if (raiz->chave == ch) return raiz;
  PONT p = raiz->primFilho;
  while(p) {
    PONT resp = buscaChave(ch, p);
    if (resp) return(resp);
    p = p->proxIrmao;
  }
  return(NULL);
}
```

Efetuamos a busca na raiz

Árvores N-árias – Busca

```
PONT buscaChave(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return NULL;
  if (raiz->chave == ch) return raiz;
  PONT p = raiz->primFilho;
  while(p) {
    PONT resp = buscaChave(ch, p);
    if (resp) return(resp);
    p = p->proxIrmao;
  }
  return(NULL);
}
```

Se não encontrarmos, olhamos os filhos

Árvores N-árias – Busca

```
PONT buscaChave(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return NULL;
  if (raiz->chave == ch) return raiz;
  PONT p = raiz->primFilho;
  while(p) {
    PONT resp = buscaChave(ch, p);
    if (resp) return(resp);
    p = p->proxIrmao;
  }
  return(NULL);
}
```

E, para cada filho, buscamos na subárvore de que ele é raiz

Árvores N-árias – Busca

```
PONT buscaChave(TIPOCHAVE ch, PONT raiz){
  if (raiz == NULL) return NULL;
  if (raiz->chave == ch) return raiz;
  PONT p = raiz->primFilho;
  while(p) {
    PONT resp = buscaChave(ch, p);
    if (resp) return(resp);
    p = p->proxIrmao;
  }
  return(NULL);
}
```

Se não encontrarmos no pai e nem nos descendentes, não está lá

Árvores N-árias

E a exclusão?

Árvores N-árias

E a exclusão? A exclusão vai depender muito do contexto em que ela será usada

Árvores N-árias

E a exclusão? A exclusão vai depender muito do contexto em que ela será usada

Pode até nem ser necessária

Árvores N-árias

E a exclusão? A exclusão vai depender muito do contexto em que ela será usada

Pode até nem ser necessária

Em especial, temos que decidir o que fazer com os nós filhos do nó excluído

Árvores N-árias

E a exclusão? A exclusão vai depender muito do contexto em que ela será usada

Pode até nem ser necessária

Em especial, temos que decidir o que fazer com os nós filhos do nó excluído

Serão adotados por alguém? Ou morrem junto?

Árvores N-árias

E a exclusão? A exclusão vai depender muito do contexto em que ela será usada

Pode até nem ser necessária

Em especial, temos que decidir o que fazer com os nós filhos do nó excluído

- Serão adotados por alguém? Ou morrem junto?
- É uma decisão de projeto... exercício para você ☺

AULA 19
ESTRUTURA DE DADOS

Árvores N-árias

Norton T. Roman & Luciano A. Digiampietri