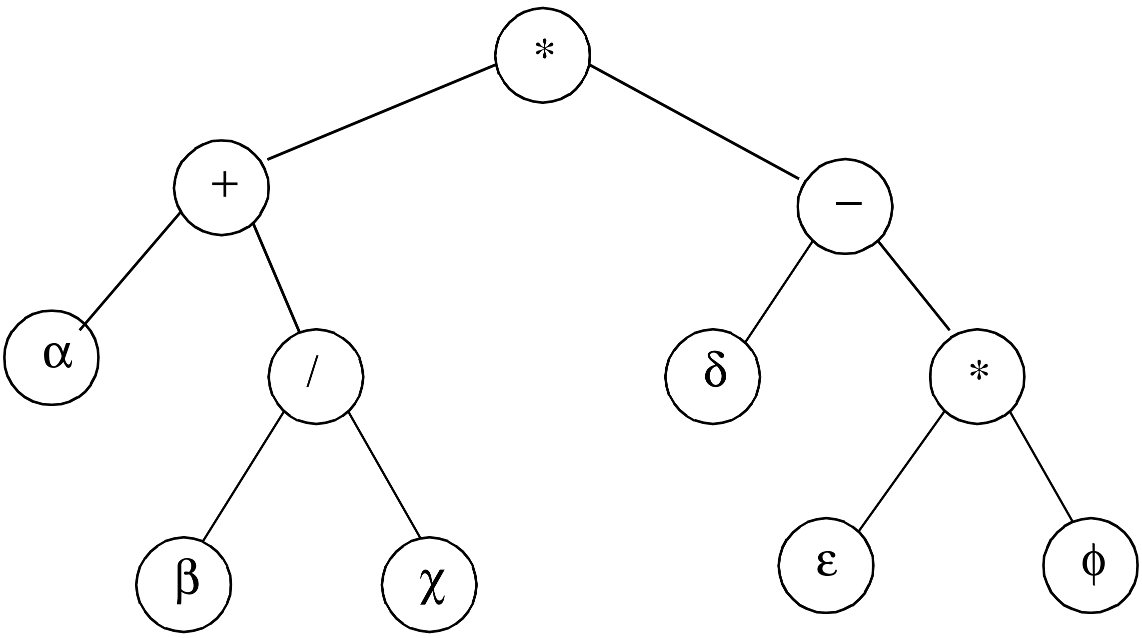
8. Árvores

8.1. Algoritmos de Busca em Árvores

Como exemplo consideremos uma forma de se representar uma expressão de operadores diádicos (binários): a árvore.

Voce poe representar qualquer expressão onde todos os operadores aceitem dois argumentos (p.ex.: uma expressão somente com as operações elementares) através de uma árvore:

- Cada operador representa uma bifurcação,
- Seus dois operandos correspondentes são representados pelas suas subárvores.
- Isto é uma forma recursiva de se representar uma expressão.



Representação da expressão $(\alpha+\beta/\chi)*(\delta-\varepsilon*\phi)$ como árvore

- Observe que a precedência de operadores foi considerada.

8.1.1. Como percorrer uma árvore binária simples

✔ Existem três ordens para se percorrer uma árvore que são conseqüência natural da estrutura da árvore:

- Preordem (r,e,d) - *Preorder*
- Emordem (e,r,d) - *Inorder*
- Pósordem (e,d,r) - *Postorder*

- Essas ordens são definidas recursivamente (definição natural para uma árvore) e em função da raiz(r), da subárvore esquerda(e) e da subárvore direita(d).

1. Preordem (r,e,d): Visite a raiz ANTES das subárvores.
2. Emordem (e,r,d): Visite a primeiro a subárvore ESQUERDA, depois a RAIZ e depois a subárvore DIREITA.
3. Pósordem (e,d,r): Visite araiz DEPOIS das subárvores.

- As subárvores são SEMPRE visitadas da esquerda para a direita.

Se percorrermos a àrvore anterior usando os algoritmos acima, teremos as seguintes seqüências:

1. Preordem: $* + \alpha / \beta \chi - \delta * \varepsilon \phi$ Notação Prefixada
2. Emordem: $\alpha + \beta / \chi * \delta - \varepsilon * \phi$ Notação Infixada*
3. Posordem: $\alpha \beta \chi / + \delta \varepsilon \phi * - *$ Notação Posfixada

onde podemos reconhecer as três formas das expressões: percurso preordem devolve a expressão em notação prefixada, emordem em notação infixada e posordem em notação posfixada, mesmo que sem os parênteses necessários.

✔ Podemos definir os três métodos como procedimentos recursivos:

```
Algoritmo Preordem (tipoArvore : *arvore)
   início
      se (arvore não é nulo) então
         imprima arvore->info
         preordem(arvore->esq)
         preordem(arvore->dir)
      fim se
   fim Preordem

Algoritmo Emordem (tipoArvore : *arvore)
   início
      se (arvore não é nulo) então
         Emordem(arvore->esq)
         imprima arvore->info
         Emordem(arvore->dir)
      fim se
   fim Emordem

Algoritmo Posordem (tipoArvore : *arvore)
   início
      se (arvore não é nulo) então
         Posordem(arvore->esq)
         Posordem(arvore->dir)
         imprima arvore->info
      fim se
   fim Posordem
```

8.1.2. Exercícios: Percurso em Árvores

1. Implemente os algoritmos de busca dados juntamente com a classe árvore dada aula anterior e teste todos os três com expressões aritméticas diferentes. Mude o campo Info para String, para que possa armazenar nomes de variáveis e operadores como DIV.

2. O algoritmo tradicional para montar uma árvore binária não seria capaz de montar uma árvore com a expressão do exemplo anterior. Para que a árvore seja montada corretamente é necessário que parênteses e a precedência de operadores sejam respeitados.
   Conceba, em linhas gerais, um algoritmo para ler uma expressão qualquer com as quatro operações e qualquer número de parênteses e as colocar em uma árvore. Utilize uma pilha como armazenamento temporário, caso necessário.

3. Os 3 algoritmos de percurso de árvores binárias são definidos e implementados recursivamente pois esta é a forma mais natural de se realizá-los, já que ela está próxima à definição destes.
   Você pode também implementar qualquer um deles de forma iterativa, utilizando para isso uma pilha para armazenar nodos da árvore para os quais voce precisa retornar ao descer na mesma.
   Escolha um dos métodos de percuros de árvore dados e reimplemente-o de forma iterativa, usando um algoritmo com um laço e uma pilha para simular a recursividade durante a descida.