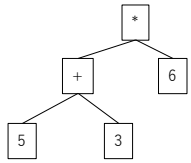
構文木の生成規則

式の構文木とは計算順序を加味した演算子と被演算子の間の構造を表すものです。
例えば式 (5+3)*6 の構文木は以下のようになります。



← 先に 5 と 3 を足し、次に 6 をかける。

構文木から逆ポーランド記法を得るには、下から上に向かって演算子と被演算子を並べていきます。

→ 5 3 + 6 *

※逆ポーランド記法の詳細はWikipediaをご参照ください。

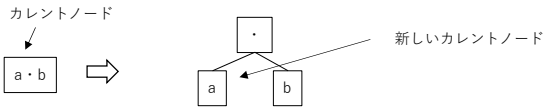
1．式の構文解析では式の文字列を後ろから読む。

2．状態遷移表は以下のとおり。

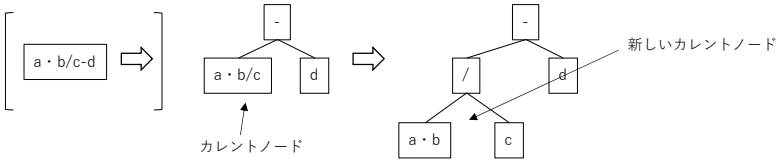
		これから読む字句								
		+	-	*	/	数字	英字	(	)	文頭
直前に読んだ字句	文末	エラー	エラー	エラー	エラー	カレントノードに格納	カレントノードに格納	エラー	レベル+	エラー
	+	エラー	エラー	エラー	エラー	カレントノードに格納	カレントノードに格納	0付加、レベル-	レベル+	0付加、正常終了
	-	エラー	エラー	エラー	エラー	カレントノードに格納	カレントノードに格納	0付加、レベル-	レベル+	0付加、正常終了
	*	エラー	エラー	エラー	エラー	カレントノードに格納	カレントノードに格納	エラー	レベル+	エラー
	/	エラー	エラー	エラー	エラー	カレントノードに格納	カレントノードに格納	エラー	レベル+	エラー
	数字	追加ルール参照	追加ルール参照	追加ルール参照	追加ルール参照	エラー	エラー	レベル-	エラー	正常終了
	英字	追加ルール参照	追加ルール参照	追加ルール参照	追加ルール参照	エラー	エラー	レベル-	エラー	正常終了
	(	追加ルール参照	追加ルール参照	追加ルール参照	追加ルール参照	エラー	エラー	レベル-	エラー	正常終了
	)	エラー	エラー	エラー	エラー	カレントノードに格納	カレントノードに格納	エラー	レベル+	エラー

・カレントノードとは
　構文解析の過程で、次の新規ノードを追加する位置にあたるノード

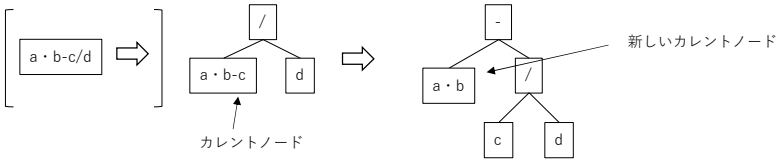
・追加ルール（カレントノードに新しいノードを追加するときのルール）
　・カレントノードの親ノードが存在しない
　　①ノードを新規作成し、読み込んだ演算子を登録。
　　②カレントノードを①のノードの右につける。
　　③空のノードを新規作成し、①のノードの左につけ、このノードを新しいカレントノードとする。



　・読み込んだ演算子がカレントノードの親ノード（演算子）より優先順位が低くない
　　①ノードを新規作成し、読み込んだ演算子を登録。
　　②カレントノードの親ノードのカレント位置に①のノードをつける。
　　③カレントノードを①のノードの右につける。
　　③空のノードを新規作成し、①のノードの左につけ、このノードを新しいカレントノードとする。



　・読み込んだ演算子がカレントノードの親ノード（演算子）より優先順位が低い
　　①ノードを新規作成し、読み込んだ演算子を登録。
　　②カレントノードの親ノードの親ノードのカレント位置に①のノードをつける。
　　③①のノードの右にカレントノードをつける。
　　④空のノードを新規作成し、①のノードの左につけ、このノードを新しいカレントノードとする。



・0付加
　カレントノードに0を登録する
　（ここでノードの生成は終了することになる）

・レベル+（計算優先を意味するカッコ( )の中に入っていくこと）
　次の読込位置から解析をスタートし、( )の中の式の構文木作成を行う。
　（プログラム上は再帰呼び出しを行う）

・レベル-（計算優先を意味するカッコ( )の中から出ること）
　解析を止める。構文木のルートノードを呼び出し元のカレントノードのカレント位置につける。

3．構文木のノードを表すクラスは以下の構成とする。

SyntaxNodeクラス

string Value	演算子や整数、piなどの字句を保持。
SyntaxNode LeftChild	ノードの左につく子ノード
SyntaxNode RightChild	ノードの右につく子ノード
SyntaxNode Parent	ノードの親ノード
SyntaxNode Root	ノードのルートノード（親ノードをさかのぼって行き着いたときのノード）