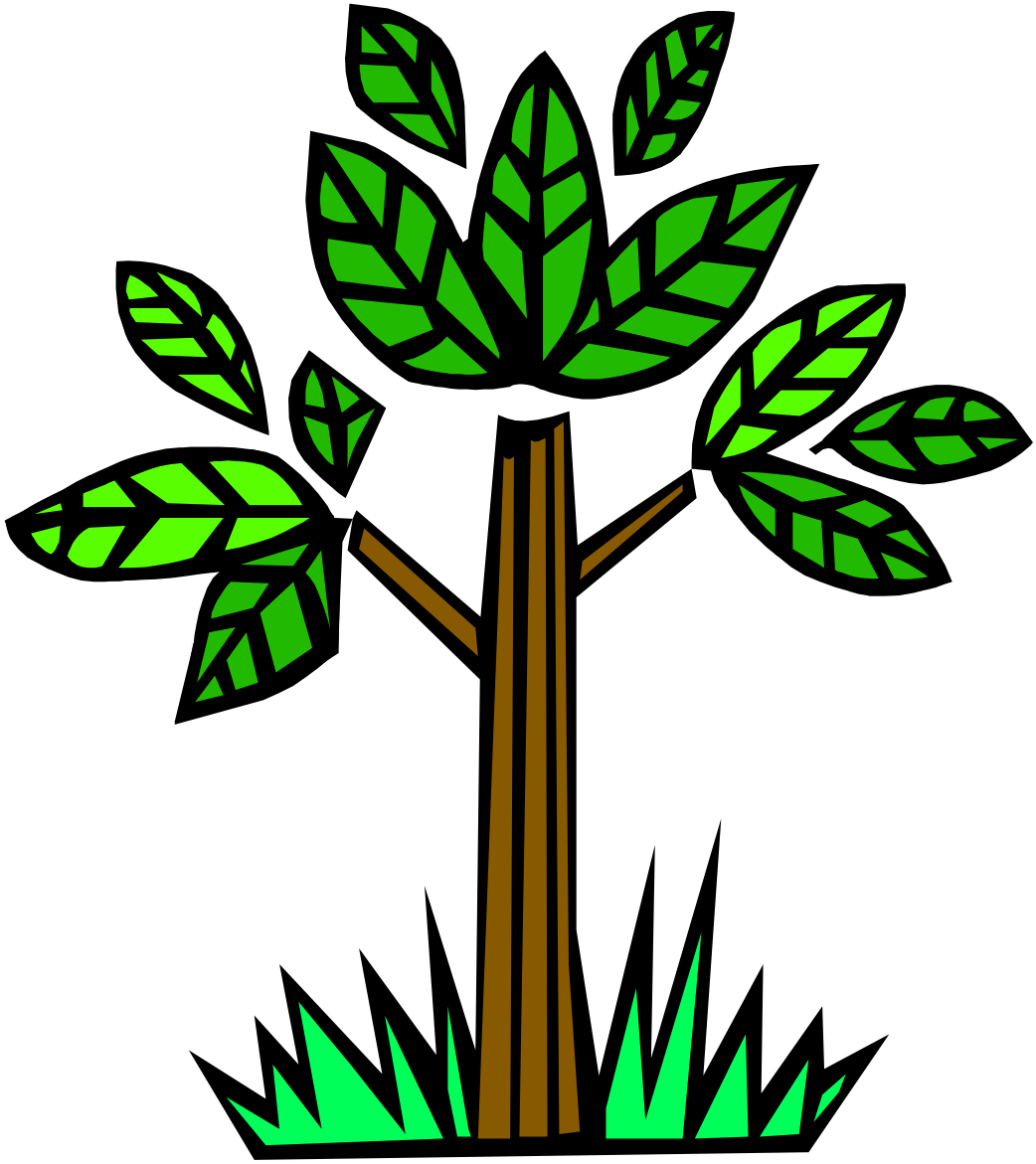
Binäre Bäume

Darstellung und Traversierung



Name	Frank Bollwig
Matrikel-Nr.	2770085
E-Mail	fb641378@inf.tu-dresden.de
Datum	15. November 2001

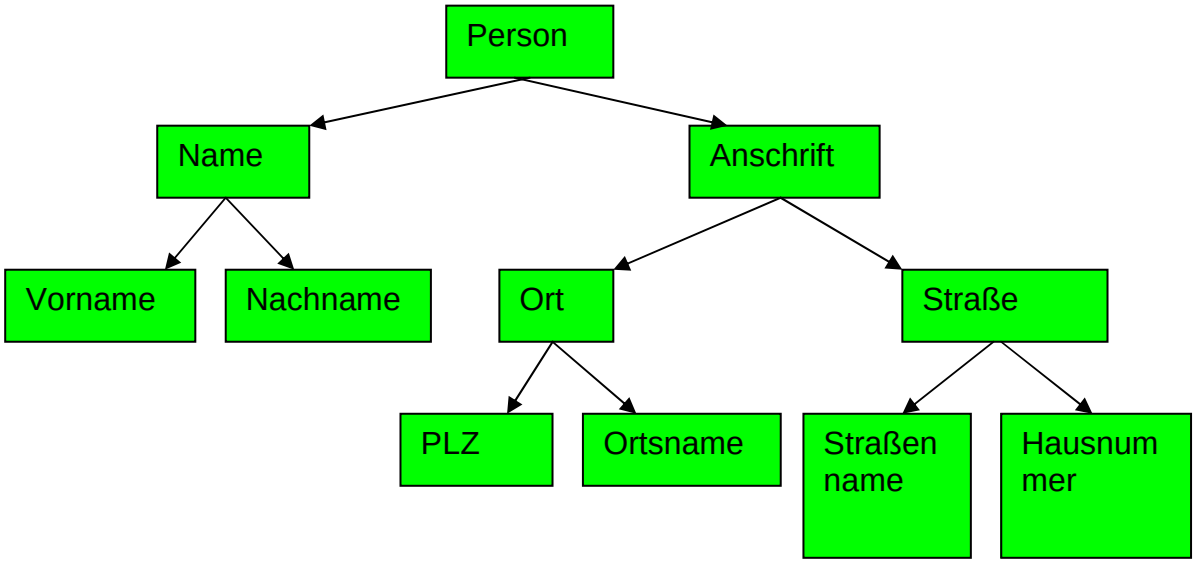
0. Vorbemerkungen 3
1. Terminologie binärer Bäume 4
2. Darstellung binärer Bäume 5
3. Traversierung binärer Bäume 9
   3.1 Inorder- Traversierung 9
   3.2 Preorder- Traversierung 9
   3.3 Postorder- Traversierung 10
   3.4 Level- Order- Traversierung 10
4. Realisierung in Logo 11
5. arithmetische Ausdrücke als binäre Bäume 13

0. Vorbemerkungen

Bäume sind zweidimensionale verkettete Strukturen und für viele Algorithmen von zentraler Bedeutung.

- **Grundgedanke:**

  - Familienbaum mit den Vorfahren
  - Ausscheidungskämpfe bei Sportturnieren
    - der Gewinner ist die Wurzel,
    - auf der 1. Ebene sind die Teilnehmer des Finales,
    - auf der 2. Ebene sind die Teilnehmer der Halbfinale,
    - auf der 3. Ebene sind die Teilnehmer der Viertelfinale,
    - etc.
  - personenbezogene Daten



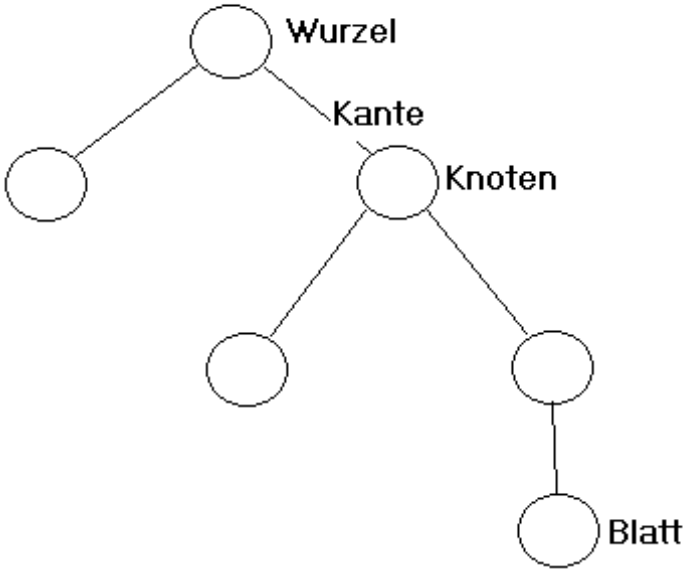
- **Informatik**

  Felder, Listen, Stapel, Schlangen und *Bäume* bilden die Grundlage für praktisch alle Algorithmen.

  Die Baumstruktur ist eines der wesentlichsten graphischen Modelle in der Datei- und Zugriffsmodellierung.

  Die wichtigste Baumstruktur ist die Binärbaumstruktur.

1. Terminologie binärer Bäume



Ein Baum ist eine nichtleere Menge von Knoten und Kanten, die gewissen Forderungen genügt:

a) **Knoten**: einfaches Objekt, das einen Namen haben und mit ihm verknüpfte Informationen tragen kann
b) **Kante**: Verbindung zwischen zwei Knoten
c) Einer der Knoten wird als **Wurzel** bezeichnet. Die Wurzel ist ein Knoten ohne einlaufende Kante.
d) Jeder Knoten (außer der Wurzel) besitzt genau einen Knoten, der sich unmittelbar über ihm befindet und als sein direkter Vorgänger (Vater) bezeichnet wird; Knoten unmittelbar unter einem Knoten werden seine direkten Nachfolger genannt (Kinder, Sohn, Geschwister) .
e) Knoten ohne Nachfolger werden **Blätter** (*Endknoten* oder *äußere Knoten*) genannt.
f) Knoten mit Nachfolger werden *Nichtendknoten* oder *innere Knoten* genannt.
g) Jeder Knoten ist die Wurzel eines Unterbaumes.
h) Ein geordneter Baum ist ein Baum, in welchem die Reihenfolge der direkten Nachfolger bei jedem Knoten angegeben ist.
i) Die Knoten eines Baumes lassen sich in **Ebenen** (Stufen) einteilen.
j) Falls jeder Knoten eine bestimmte Anzahl von Nachfolgern haben muß, die in einer bestimmten Reihenfolge erscheinen, so haben wir es mit **n- nären Bäumen** zu tun.

Der binäre Baum ist ein geordneter Baum, der aus zwei Typen von Knoten besteht:
äußere Knoten (Knoten ohne Nachfolger),
innere Knoten (Knoten mit Nachfolger).
Da die zwei direkten Nachfolger eines jeden inneren Knoten geordnet sind, sprechen wir vom linken und rechten Nachfolger innerer Knoten.

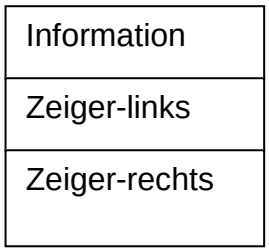
Operationen auf binären Bäumen:

- Anlegen eines Elements
- Zugriff auf das erste Element eines Baumes
- Durchlaufen (Traversieren) des Baumes zum Bearbeiten von Elementen
- Suchen eines Elements
- Einfügen im Baum
- Löschen im Baum

2. Darstellung binärer Bäume

Die gebräuchlichste Darstellung von binären Bäumen ist eine einfache Benutzung von Datensätzen mit zwei Verkettungen.

Datenstruktur:



Es gibt einen Zeiger auf die Wurzel; wenn der Zeiger *NIL* ist, dann ist der binäre Baum leer.

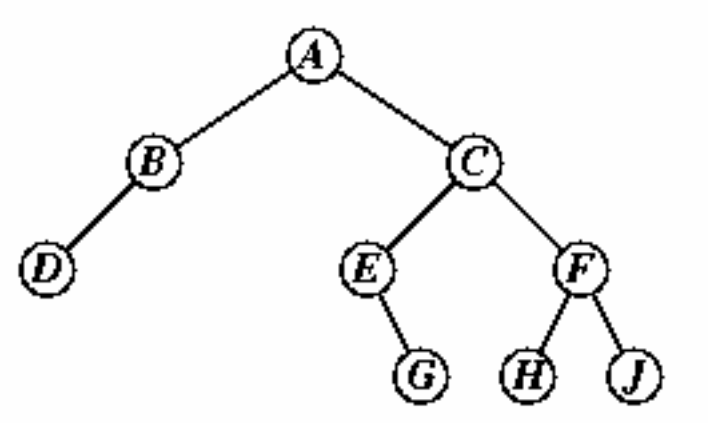
Ein Knoten des binären Baumes besteht aus der ihm zugeordneten Information und den beiden Zeigern auf den linken und rechten Teilbaum.

Quelltext zu Delphi

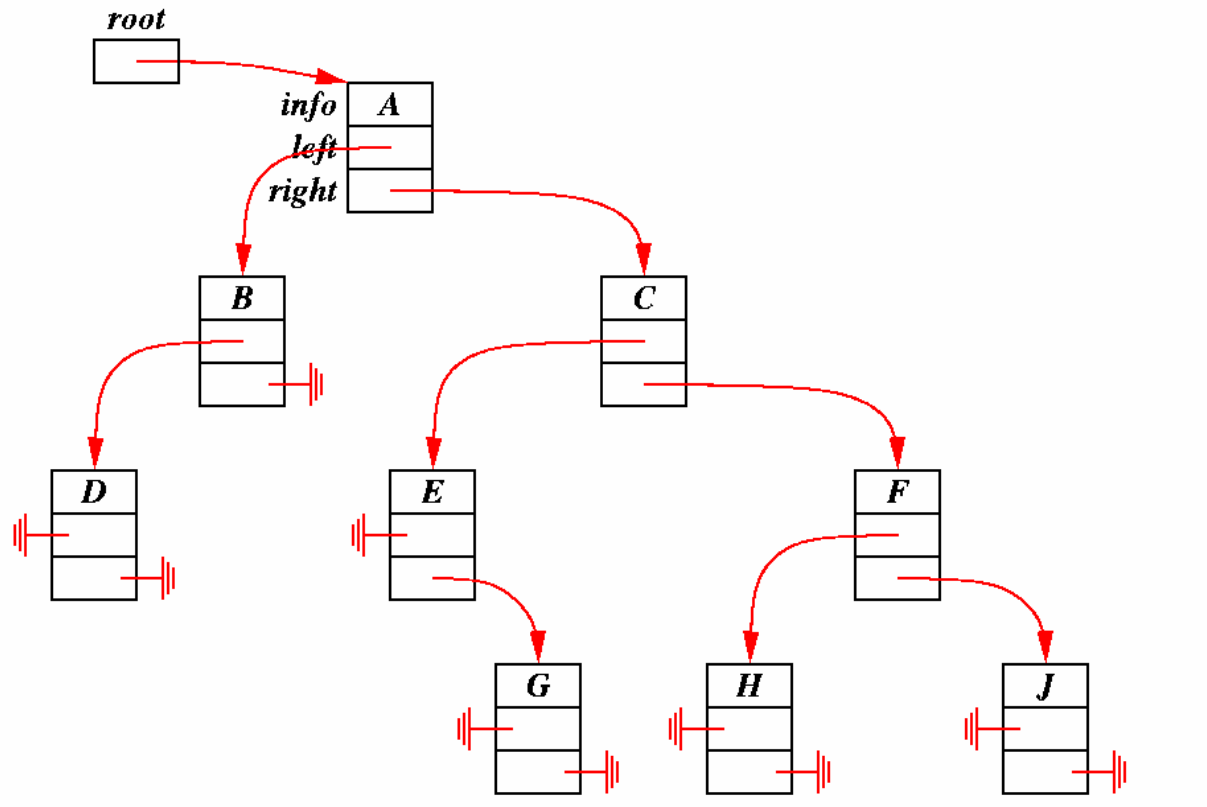
```
Type zeiger =^Knoten;

Knoten = Record
            Inhalt : integer;
            Links : zeiger;
            Rechts : zeiger;
End;
```

Beispiel:



So sieht die Repräsentierung des obigen Baumes aus.

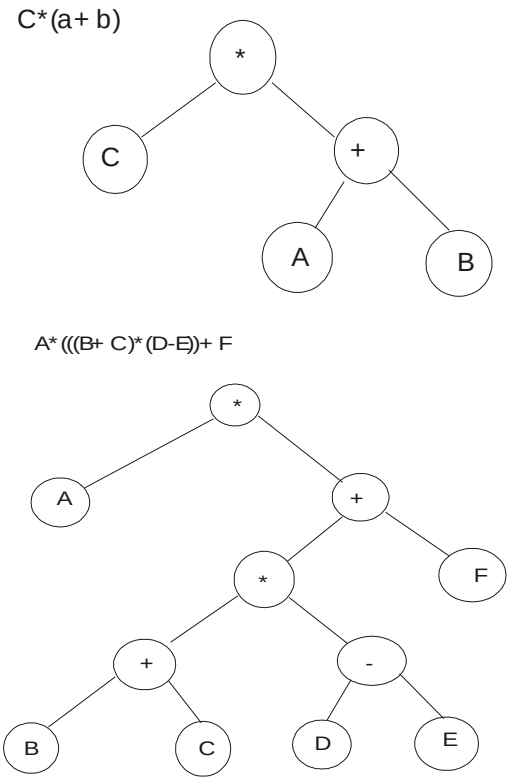


Beispiel: Verarbeitung arithmetischer Ausdrücke (Syntaxbäume)

Der Syntaxbaum für einen arithmetischen Ausdruck wird durch folgende rekursive Regel bestimmt:

1. Man setze den Operator an die Wurzel.
2. Man setze den Baum für den Ausdruck, der dem ersten Operanden entspricht, auf die linke Seite.
3. Man setze den Baum für den Ausdruck, der dem zweiten Operanden entspricht, auf die rechte Seite.

In den inneren Knoten sind die Operatoren, in den Blättern werden die Argumente gespeichert.



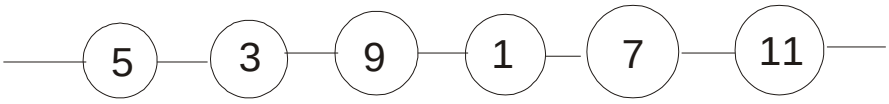
Da die Operatoren genau zwei Operanden verwenden, ist ein binärer Baum für diese Art von Ausdrücken geeignet. (Kompliziertere Ausdrücke können einen anderen Baumtyp erfordern.)

Beispiel : Darstellung von Mengen

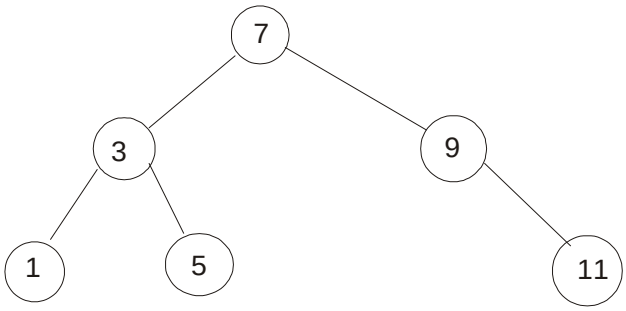
Jeder Knoten des binären Baumes enthält ein Element der Menge und zwei Verweise auf – möglicherweise leere – Knoten.
Der linke Zweig verweist auf kleinere Elemente, der rechte auf größere Elemente.

$\{1,3,5,7,9,11\}$

Darstellung als lineare Liste:



Darstellung als binärer Baum:



Test, ob eine Zahl x in einer Menge enthalten ist.

- Vergleich von x mit dem Wurzelknoten des Baumes
- Ist x kleiner als die Wurzel, wird im linken Teilbaum weitergesucht
- Ist x größer als die Wurzel, wird im rechten Teilbaum weitergesucht

3. Traversierung binärer Bäume

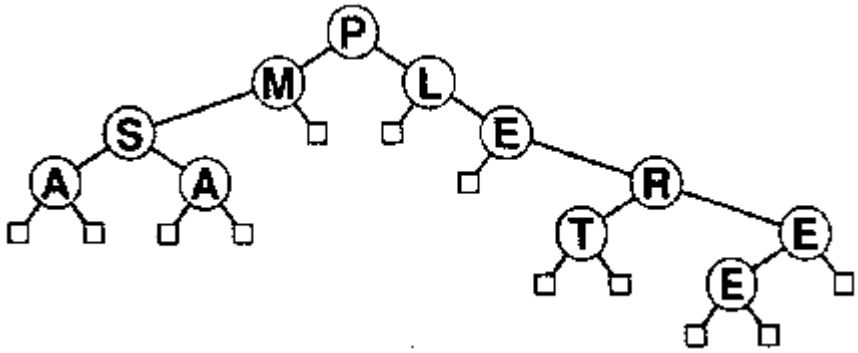
Unter der Traversierung von Bäumen versteht man das systematische Besuchen jeden Knotens.
Da man auf die einzelnen Knoten eines Baumes nur über die Wurzel zugreifen kann, benötigt man Algorithmen, die beim Durchlaufen des Baumes jeden Knoten genau einmal besuchen und auswerten.
Dabei kann man vier Methoden unterscheiden, wenn der Baum nicht leer ist:

1. Inorder- Traversierung
2. Preorder- Traversierung
3. Postorder- Traversierung
4. Level- Order- Traversierung

3.1 Inorder- Traversierung

Die Inorder- Traversierung ist wahrscheinlich am weitesten verbreitet.
Rekursive Regel: **LWR**

1. Besuche den linken Unterbaum.
2. Besuche die Wurzel.
3. Besuche den rechten Unterbaum

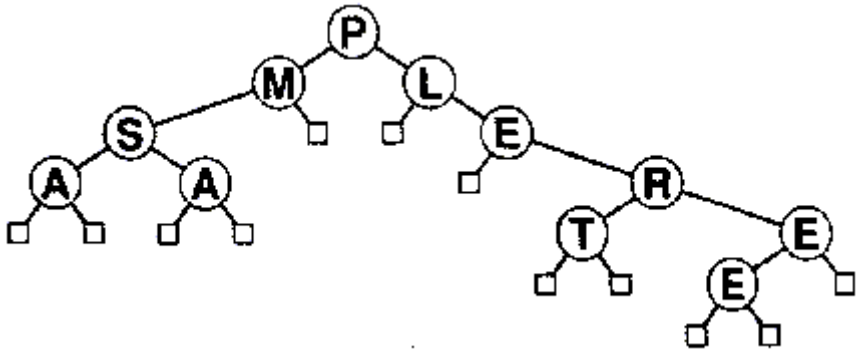


Inorder-Durchlauf: A S A M P L E T R E E

3.2 Preorder- Traversierung

Rekursive Regel: **WLR**

1. Besuche die Wurzel.
2. Besuche den linken Unterbaum.
3. Besuche den rechten Unterbaum.

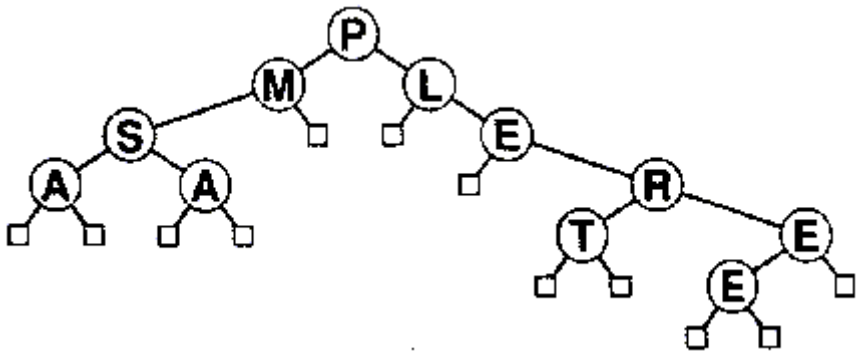


Preorder-Durchlauf: P M S A A L E R T E E

3.3 Postorder- Traversierung

rekursive Regel: **LRW**

1. Besuche den linken Unterbaum.
2. Besuche den rechten Unterbaum.
3. Besuche die Wurzel.

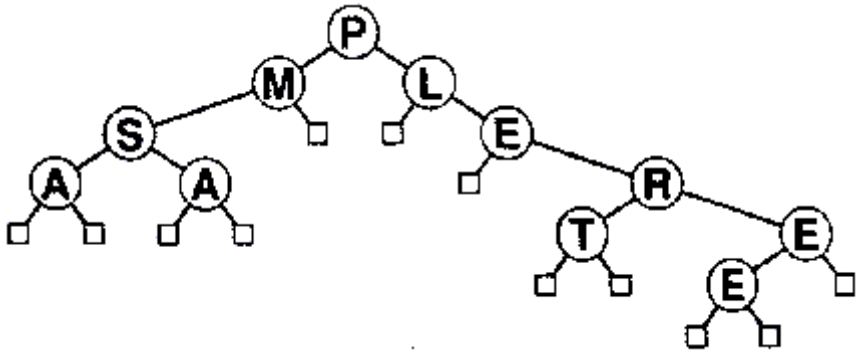


Postorder-Durchlauf: A A S M T E E R E L P

3.4 Level- Order- Traversierung

nicht rekursiv

Regel: Besuche die Knoten so, wie sie in der Abbildung erscheinen, d.h. sie werden von oben nach unten und von links nach rechts gelesen.



Level-Order-Durchlauf: P M L S E A A R T E E

Die vier Methoden unterscheiden sich, weil die Reihenfolge, in der die Knoten besucht werden, verschieden ist; d.h. je nach Anwendung ist eine bestimmte Traversierung des Baumes zweckmäßig.

4. Realisierung in Logo

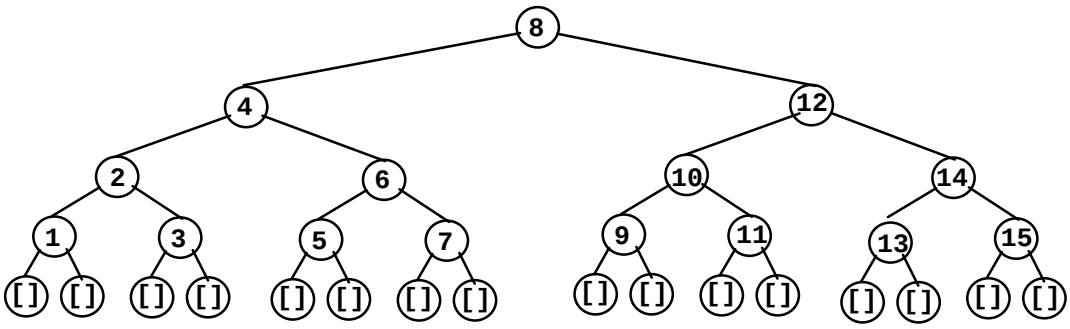
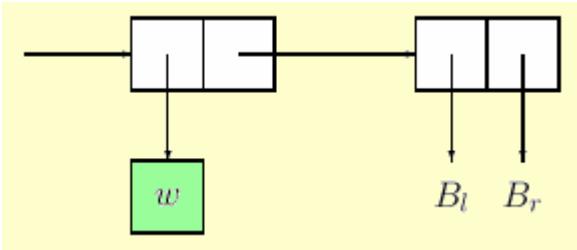


Abb. : Binärer Baum

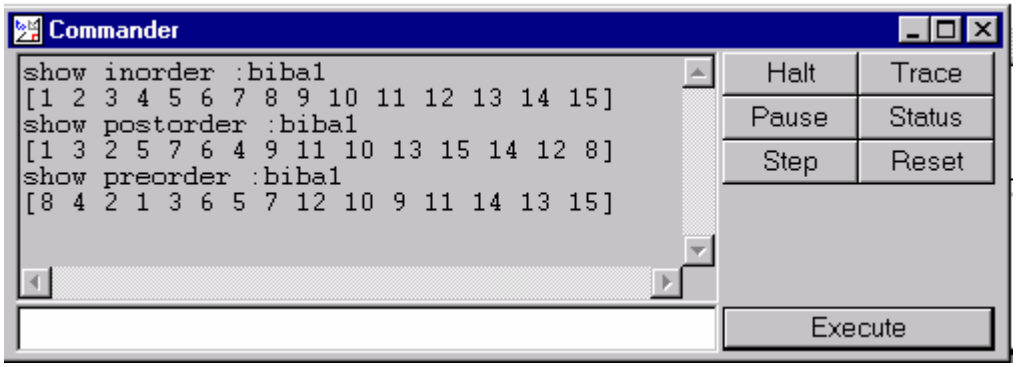



```
to baumdefinition
 Make "biba1 [8  [4  [2  [1 ] [3  ]] [6  [5  ] [7  ]]]~
               [12 [10 [9 ] [11 ]] [14 [13 ] [15 ]]]]
end
```

```
to postorder :b
; Funktion, die die Knoten in Postorder-Reihenfolge als Liste
zurueckgibt
 ifelse emptyp :b [op :b] ~
                  [op (sentence postorder first bf :b ~
                                postorder first bf bf :b ~
                                first :b)]
end


to inorder :b
; Prozedur, die die Knoten in Inorder-Reihenfolge als Liste
zurueckgibt
 ifelse emptyp :b [op :b] ~
                  [op (sentence inorder first bf :b ~
                                first :b  ~
                                inorder first bf bf :b)]
end

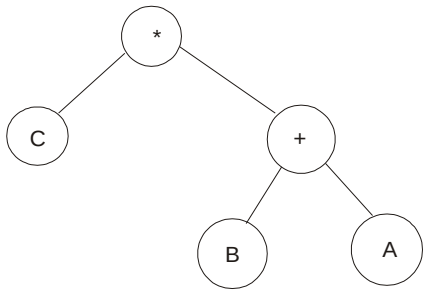
to preorder :b
; Funktion, die die Knoten in Preorder-Reihenfolge als Liste
zurueckgibt
 ifelse emptyp :b [op :b] ~
                  [op (sentence first :b ~
                                preorder first bf :b ~
                                preorder first bf bf :b)]
end
```



5. arithmetische Ausdrücke als binäre Bäume

Operatoren bilden die inneren Knoten, die äußeren Knoten werden für die Variablen verwendet.
Unitäre Operatoren haben nur einen Teilbaum, wobei hierfür der rechte genommen wird.

Beispiel : C*(B+A)

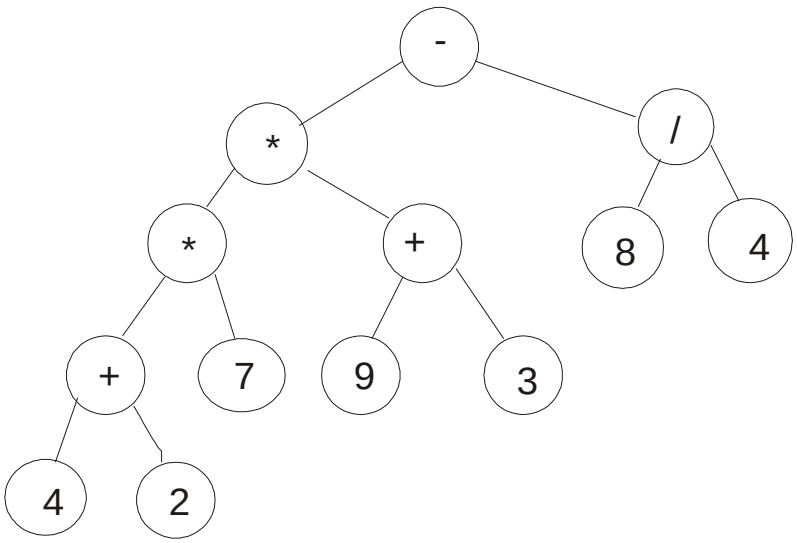


Traversierung

1.	Inorder	LWR	C*B+A	Infix- Notation
2.	Preorder	WLR	*C+BA	Präfix- Notation
3.	Postorder	LRW	CBA+*	Postfix- Notation

Beispiel: ((4+2)*7*(9+3)) - (8/4)

Darstellung als binärer Baum



Traversierung

1.	Inorder	LWR	4+2*7*9+3-8/4	Infix- Notation
2.	Preorder	WLR	-**+427+93/84	Präfix- Notation
3.	Postorder	LRW	42+7*93+*84/-	Postfix- Notation

Bei der Infix- Notation ergibt sich durch die Klammern ein zusätzlicher Aufwand. Die Postfix- Notation kommt ohne Klammern aus.

Die Notation der Preorder- und Postorder- Traversen wurden zuerst von dem polnischen Logiker Jan Lukasiewicz eingeführt. Entsprechend wird die Preorder- Traverse als polnische Notation und die Postorder- Traverse als umgekehrte polnische Notation bezeichnet.