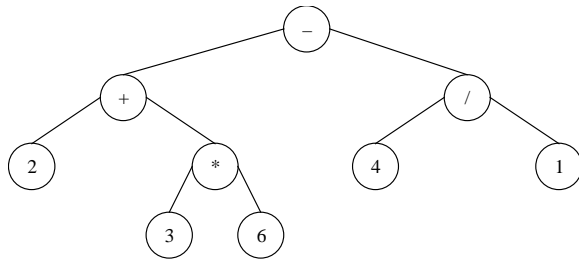


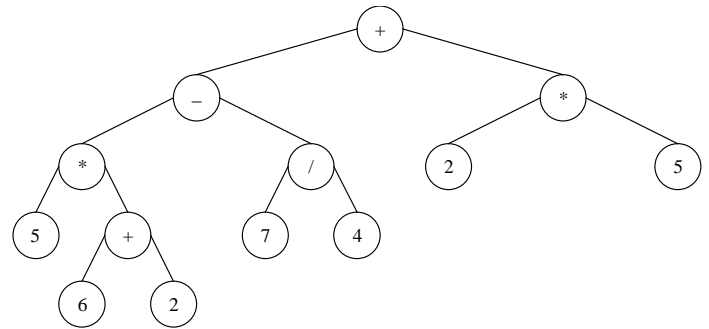
Aufgabe: *Termbäume*

In der fünften Klasse lernt man, nach welchen Rechengesetzen Terme berechnet werden: Klammer- vor Punkt- und Punkt- vor Strichrechnung. Zur Veranschaulichung werden sogenannte Rechenbäume oder *Termbäume* benutzt, welche die Reihenfolge der Berechnung veranschaulichen. Zwei Beispiele:

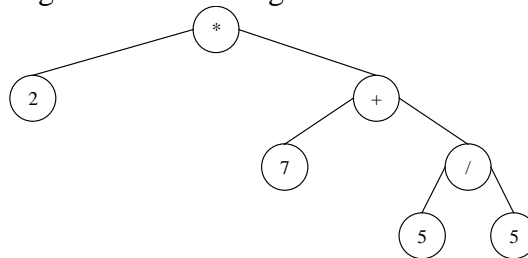
Beispiel 1: $2 + 3 * 6 - 4 / 1$



Beispiel 2: $5 * (6 + 2) - 7 / 4 + 2 * 5$



a) Stellen Sie den zugehörigen Term zum folgenden Termbaum auf:



- b) Stellen Sie den zugehörigen Termbaum zum Term $7 * 6 - (3 * 5 - 2 + 1)$ auf.
- c) Analysieren Sie die Baumstruktur der Termbäume. Welche Eigenschaften fallen Ihnen dabei auf?
- d) Durchläuft man einen Termbaum in Preorder, so erhält man den sogenannten *Präfix-Term*, durchläuft man ihn in Postorder, so erhält man den *Postfix-Term*. Geben Sie Präfix-Term und Postfix-Term des Termbaums aus **Beispiel 1** an.
- e) Geben Sie den Termbaum zum Präfix-Term $- * 2 + 2 3 + 6 1$ an.
- f) Schreiben Sie eine Prozedur, welche aus einem Präfix-Term den zugehörigen Termbaum aufstellt. Nutzen Sie dabei das Programmraaster auf der vorbereiteten Lösungsfolie (**Anlage I**) sowie das im Unterricht entwickelte Tool zum Binärbaum (**Anlage II**).

Hinweis: Sollten Sie die Prozedur nicht entwickeln können, so erläutern Sie stattdessen ihre algorithmische Idee.

Viel Erfolg!

Anlage I Lösungsfolie für Aufgabenteil f)

```
uses B_Baum;
```

```
const Operatoren: set of char= ['+', '-', '*', '/'];  
      Operanden: set of char= ['0'..'9'];
```

```
function NaechstesZeichen(var Zeichenkette: string): char;  
begin  
  NaechstesZeichen:= Zeichenkette[1];  
  Delete(Zeichenkette,1,1);  
end;
```

```
procedure ErstelleTermBaumAusPraefixterm(var Termbaum: PBinBaum; var Praefixterm: string);  
var
```

```
begin
```

```
end;
```

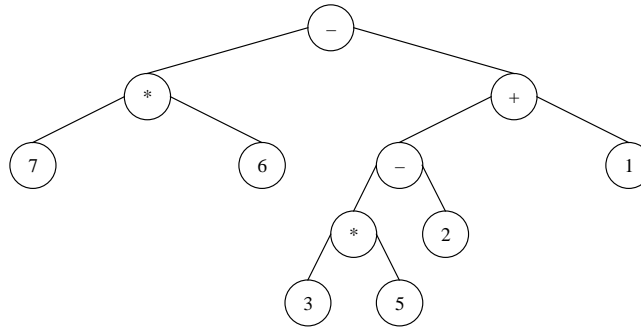
Anlage II Im Unterricht entwickeltes Tool zum Datentyp Binärbaum

```
-----
{ TOOL zum Datentyp Binärbaum: - Filenamen: <B_BAUM.ADT> }
{ <B_BAUM.TPU> }
{ Es wird der Datentyp 'PBinBaum' bereitgestellt - einschließlich der für ihn }
{ definierten Operationen, abhängig vom konkreten Datentyp TInhalt !!! }
{
{ Name des Datentyps: PBinBaum [ Baum = leer oder ] }
{ [ Wurzel + linker Teilbaum ] }
{ [ + rechter Teilbaum, ] }
{ [ wobei die Teilbäume u.U. auch ] }
{ mit den Operationen: [ leer sein können (Blatt!) ] }
{
{ PROCEDURE InitBaum ( VAR baum: PBinBaum ); }
{ [ Einrichtung als leerer Baum (Initialisierung) ] }
{ FUNCTION BaumLeer ( baum: PBinBaum ): Boolean; }
{ [ ergibt TRUE gdw. der aktuelle Baum leer ist. ] }
{ FUNCTION Wurzel ( baum: PBinBaum ): TInhalt }
{ [ der Wurzelinhalt eines nicht (!) leeren Baumes wird ] }
{ [ der Inhaltsvariablen zugewiesen. ] }
{ PROCEDURE HoleLinkenTeilbaum ( baum: PBinBaum; VAR teilbaum: PBinBaum ); }
{ PROCEDURE HoleRechtenTeilbaum ( baum: PBinBaum; VAR teilbaum: PBinBaum ); }
{ [ Von einem nicht (!) leeren Baum wird der Teilbaum- ] }
{ [ variablen der bezeichnete Teilbaum zugewiesen. Der ] }
{ [ Baum selbst bleibt davon unberührt. ] }
{ PROCEDURE BelegeWurzelinhalt ( VAR baum: PBinBaum; inhalt: TInhalt ); }
{ [ einem nicht (!) leeren Baum wird der Wert der In- ] }
{ [ haltsvariablen in die Wurzel geschrieben. Die Teil- ] }
{ [ bäume bleiben davon unberührt. ] }
{ PROCEDURE FuegeTeilbaumLinksAn ( VAR baum: PBinBaum; }
{ teilbaum: PBinBaum ); }
{ PROCEDURE FuegeTeilbaumRechtsAn ( VAR baum: PBinBaum; }
{ teilbaum: PBinBaum ); }
{ [ einem nicht (!) leeren Baum wird der genannte Teil- ] }
{ [ baum angehängt. Die Teilbäume selbst müssen einge- ] }
{ [ richtet sein, können allerdings auch leer sein. Die ] }
{ [ Wurzel bleibt davon unberührt. ] }
{ PROCEDURE ErzeugeBlatt ( VAR baum: PBinBaum; inhalt: TInhalt ); }
{ [ es wird ein Baum mit dem Wurzelwert der Inhaltsvari- ] }
{ [ ablen mit leeren Teilbäumen eingerichtet. ] }
{ FUNCTION LinksLeer ( baum: PBinBaum ): Boolean; }
{ FUNCTION RechtsLeer ( baum: PBinBaum ): Boolean; }
{ [ TRUE, falls entsprechende Teilbäume leer sind. ] }
{ FUNCTION IstBlatt ( baum: PBinBaum ): BOOLEAN; }
{ [ TRUE, falls der nicht (!) leere Baum nur leere Teil- ] }
{ [ bäume besitzt. ] }
{ PROCEDURE LoescheBlatt ( VAR baum: PBinBaum ); }
{ [ ein eingerichteter und nicht (!) leerer Baum mit ] }
{ [ leeren Teilbäumen wird gelöscht, d.h. er wird leer. ] }
{ [ Außerdem wird der benutzte Speicherraum wieder dem ] }
{ [ System zur Verfügung gestellt. ] }
{ PROCEDURE LoescheBaum ( VAR baum: PBinBaum ); }
{ [ ein eingerichteter Baum wird gelöscht. Ist der Baum ] }
{ [ schon leer, so ist diese Operation ohne Funktion. ] }
{ [ ACHTUNG: auch die Teilbäume werden gelöscht ! ] }
{ PROCEDURE ErsetzeDurchLinkenTeilbaum ( VAR baum: PBinBaum ); }
{ PROCEDURE ErsetzeDurchRechtenTeilbaum ( VAR baum: PBinBaum ); }
{ [ der jeweils andere Teilbaum wird, falls nötig, ge- ] }
{ [ löscht und 'baum' wird ersetzt. ] }
{ ----- Version 1.0 D.Garmann / Bonn, am 22.02.2000 }
}
```

Lösungen: Termbäume

a) Der Term lautet: $2 * (7 + 5 / 5)$

b) Der Termbaum sieht wie folgt aus:
(Eventuelle Fehlerquelle beim rechten Teilbaum: Baum um den Knoten (+) rechts gedreht)

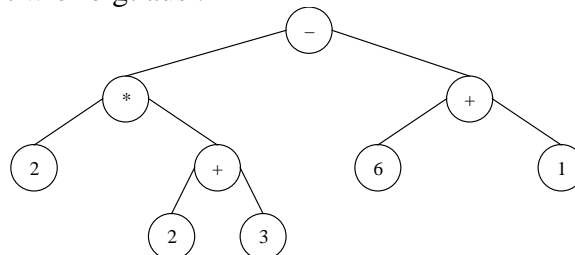


c) Die Besonderheiten sind:

- 1) Jeder Knoten ist entweder Blatt oder hat zwei nichtleere Teilbäume
- 2) Ein Baum ist ein Blatt \Leftrightarrow Wurzelinhalt ist Operand
- 3) daraus folgt: Ein Baum ist kein Blatt \Leftrightarrow Wurzelinhalt ist Operator

d) Präfix: $- + 2 * 3 6 / 4 1$, Postfix: $2 3 6 * + 4 1 / -$

e) Der Termbaum sieht wie folgt aus :



[Die zugehörige Infix-Notation wäre: $2 * (2 + 3) - (6 + 1)$]

f) Die geforderte Prozedur könnte z. B. wie folgt implementiert sein:

```

procedure ErstelleTermBaumAusPraefix(var Termbaum: PBinBaum; var Praefixterm: string);
var ltb, rtb: PBinBaum;
    Zeichen: char;
begin
    InitBinBaum(ltb);
    InitBinBaum(rtb);
    Zeichen:= NaechstesZeichen(Praefixterm);
    ErzeugeBlatt(Termbaum, Zeichen);

    if Zeichen in Operatoren
    then begin
        ErstelleTermBaumAusPraefix(ltb,Zeichenkette);
        FuegeTeilbaumLinksAn(Termbaum,ltb);
        ErstelleTermBaumAusPraefix(rtb,Zeichenkette);
        FuegeTeilbaumRechtsAn(Termbaum,rtb);
    end;
end;

```

{ Es würde nur eine Variable reichen. }

{ akt. bearbeitetes Zeichen des Terms }

{ Linken Teilbaum initialisieren. }

{ Rechten Teilbaum initialisieren. }

{ Nächstes Zeichen holen. }

{ Aus dem Zeichen ein Blatt machen. }

{ Ist das Zeichen ein Operand, dann }

{ bleibt der Baum ein Blatt, sonst }

{ ist es ein Operator und man muss }

{ aus dem Rest der Zeichenkette }

{ den linken Teilbaum erstellen }

{ und an der Operator anhängen, }

{ und genauso den rechten Tbaum }

{ erstellen und anhängen. }

Sollte die Prozedur nicht korrekt sein, so sollte wenigstens die algorithmische Idee erläutert werden.

Aufstellung der Unterrichtsinhalte

Kurshalbjahr	Inhalte
12 I	<ol style="list-style-type: none"> 1. Turtlegrafik (Sierpinski, Pythagorasbaum) 2. Höhere Sortierverfahren (Quick- Merge- Heapsort) und Aufwandsbetrachtung 3. DELPHI-Pascal-Unterschiede am Projekt Spielautomat/TR/Lotto 4. Backtracking: n-Damen 5. sequentielle Dateien: ADT intern, extern 6. einfach verkettete Listen und deren Grundoperationen, Sonderformen: Schlange
12 II	<ol style="list-style-type: none"> 1. Der Binärbaum und seine Grundoperationen (Expertensystem, Suchbaum), Traversierungen 2. Aufbau und Arbeitsweise von DV-Anlagen (geschichtl. Entwicklung, v. Neumann-Konzept) 3. Reduktion von der Hochsprache zur Maschinenebene (Pascal – RePascal – ALI-Modellassembler – Sprache der Modellmaschine): Abbildung von Kontrollstrukturen, Datenstrukturen
13 I	<ol style="list-style-type: none"> 1. Fortsetzung Reduktion von der Hochsprache zur Maschinenebene: Abbildung von Gliederungsstrukturen (Prozeduren/Funktionen mit Variablenübergabe CbR/CbV) 2. Compilerbau: formale Sprachen: kontextfreie Grammatiken (Parser), endliche Automaten (Scanner) und diesbezügliche Anwendungen, Projekt: Implementierung eines Compilers. 3. Einführung in C++: Kontroll-, Daten- und Gliederungsstrukturen
13 II	<ol style="list-style-type: none"> 1. OOP: Datenunabhängige einfach verkettete Listen (Schlange, Stack), Sortierverfahren 2. Baumstrukturen: AVL-Bäume

Die Aufgabe stammt im wesentlichen aus dem Kurshalbjahr 12 II, wobei die Termanalyse eher Bestandteil des Kurshalbjahres 13 I war. Die Schüler haben Binärbäume – auch in der besonderen Struktur der Termbäume am Beispiel Expertensystem – kennen gelernt und haben die in den Aufgaben geforderten Traversierungen durchgeführt und implementiert.

Der Anwendungszusammenhang der Binärbäume ist den Schülern allerdings nicht bekannt und erfordert so eine selbständige Leistung der Schüler. Der wohl schwierigste Aufgabenteil f) ist andeutungsweise beim Expertensystem unterrichtlich behandelt worden, allerdings wurde nie ein Algorithmus für die Herstellung eines Baumes aus der Preorderdarstellung entwickelt.

Sollte der Algorithmus aus Aufgabenteil f) nicht in DELPHI-Notation erbracht werden, so soll der Schüler auf jeden Fall die Gelegenheit bekommen, seine algorithmische Idee verbal zu formulieren.